APPLYING WEB-BASED VR TECHNOLOGIES TO SUPPORT DESIGN AND MANUFACTURING

LI JIN MSc

A thesis submitted in partial fulfilment of the
requirements of the University of Wolverhampton
for the degree of Doctor of Philosophy

April 2004

This work or any part thereof has not previously been presented in any form to the University or to any other body whether for the purposes of assessment, publication or for any other purpose (unless previously indicated). Save for any express acknowledgements, references and/or bibliographies cited in the work, I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

Signature .....

Date .....19/04/2004.....

Acknowledgments

I highly appreciate the assistance of my supervisors, Dr. Ilias A. Oraifige, Prof. Frank R. Hall and Dr. Paul M. Lister of the School of Engineering and the Built Environment, University of Wolverhampton for providing invaluable support and help in guiding and assisting me with this research project. Furthermore, many thanks to the staff and the technicians who gave me help and assistance.

And lastly, my sincere thanks to my husband and parents for their kind care and encouragement throughout the period of my PhD study in the School of Engineering and the Built Environment, University of Wolverhampton, UK.

Abstract

Virtual Reality (VR) has been applied to a wide range of industrial applications for enhanced visualization, evaluation and interactive virtual operations. Virtual Design and Manufacturing (VDM) makes VR no longer state-of-the art but rather innovative technology to support modern industry by providing interactive simulation environments to reduce risk, cost and time in the product development. With the rapid growth of network technology and the development of Web-based VR techniques, there is a potential opportunity to extend low-cost VR applications and distributed systems though the Web for e-service. The aim of this research project is to propose an approach to applying Web-based VR technologies to support design and manufacturing applications in Networked Virtual Environments (Net-VEs). This thesis proposes a cost-effective approach to applying Web-based VR to support collaborative design and manufacturing via the Internet. Based on the proposed approach, a distributed VR system has been designed and developed by the author using VRML. Furthermore, a prototype software system — the distributed VRML-based Factory Layout Simulator (VFLS) for factory layout applications has been successfully developed and implemented as a demonstrator to validate the viability of the proposed approach and the usability of distributed VR system. It is capable of aiding multiple users, especially SMEs to carry out engineering activities in a networked virtual environment at lower cost and in less time, as well as, integrating with database and information systems, sharing design and manufacturing resources through the WWW at reduced cost. It is believed that such a sharing virtual environment based upon WWW will allow industries to utilise VR sufficiently and cost-effectively for maximum impact to improve the competitiveness.

Glossary

AI	Artificial Intelligent
AOIM	Area of Interest Manager
API	Application Programming Interface
AR	Augmented Reality
BBC	British Broadcasting Corporation
BSP	Binary Space Partition
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CNC	Computer Numerical Control
CPC	Collaborative Product Commerce
CRT	Cathode Ray Tube
CSG	Constructive Solid Geometry
CVEs	Collaboration Virtual Environments
DIS	Distributed Interactive Simulation
DIVE	Distributed Interactive Virtual Environment
EAI	External Authoring Interface
FEMs	Finite Element Models
FSM	Finite State Machine
FTP	File Transfer Protocol
GUI	Graphical User Interface
HCI	Human Computer Interaction
HMDs	Head Mounted Displays
HST	Hubble Space Telescope
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPDC	Innovative Product Development Centre
IPMI	IP Multicast Initiative

ISO	International Standard Organization/
JSAI	JavaScript Application Interface
LAN	Local Area Network
LCD	Liquid Crystal Display
LOD	Level of Details
LSVES	Large-scale Virtual Environments
MASSIVE	Model, Architecture and System for Spatial Interaction in Virtual Environments
MBone	Multicast Backbone
MERL	Mitsubishi Electric Research Laboratories
MPEG-4	Motion Picture Expert Group - 4
NCSA	National Centre for Supercomputing Applications
Net-VEs	Networked Virtual Environments
NIST	National Institute of Standards and Technology
NPSNET	Naval Postgraduate School Networking
NRG	NPSNET Research Group
PBDs	Projection-based Displays
PBM	Physical Based Modelling
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transfer Protocol
RTSP	Real Time Streaming Protocol
SGI	Silicon Graphics Inc.
SIMNET	Simulator Networking
SMEs	Small and Medium-sized Enterprises
SPLINE	Scalable Platform for Large Interactive Networked Environments
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Unified Modelling Language
URL	Universal Resource Locators
US EPA	United States Environmental Protection Agency
VMD	Virtual Machine Database
VNC	Virtual Network Computing

VR	Virtual Reality
VRML	Virtual Reality Modelling Language
VRML-NG	VRML Next Generation
WWW	World Wide Web
X3D	Extensible 3D
XML	Extensible Markup Language

Table of Contents

List of Figures and Tables

Figures:

Chapter 4

Table:

Chapter 1

Introduction

1.1 General

This chapter introduces the motivation and challenges in the subject of this research work and presents the main research aims and objectives. It presents the main contributions of the author and the structure of the thesis.

1.2 Motivation and Challenges

Virtual Reality (VR) has been applied to a wide range of industrial applications associated with design and manufacturing. It has shown the powerful potential of VR in the latest generations of CAD/CAM tools to aid product development for enhanced visualization, evaluation and interactive virtual operations. The Virtual Design and Manufacturing (VDM) system is based on a virtual environment where a virtual design and manufacturing world can be modelled, simulated, visualised, and interacted with, even using devices such as Head Mounted Displays (HMDs), datagloves, and force/touch feedback accessories. The VDM has made VR no longer state-of-the art but rather an innovative technology to support modern industry by providing interactive simulation environments to reduce risk, cost and time. Engineers have applied VDM in a wide range of design

and manufacturing applications: product design and prototyping, factory layout design and visualisation, assembly process planning and simulations, machinist training, and so forth.

Product development is a complex process and therefore requires a large sophisticated information and database system in order to create and represent complex scenarios as realistically as possible in a virtual design and manufacturing environment. No doubt the management and maintenance of such large database are difficult tasks for users, especially for Small and Medium-sized Enterprises (SMEs). Another problem is the platform dependence of various VDM systems which results in incompatibility problems. For example, Deneb's software packages of integrated manufacturing solutions run on Unix platform, UGS's Solid Edge is a professional CAD system for mechanical design on Windows platform. In addition, various 3D graphics data formats also create a barrier to collaboration in digital design and manufacturing.

In this millennium, manufacturers demand to get their products to their customers faster. SMEs take the stronger competitive pressures because of the limited equipment, funds and technical capabilities. For example, Petford Tools Ltd., a small company with 29 employees in West Midlands, UK, has mainly manufactured plastic mould tools and foam tools since 1970 (Sweet, 2001). In order to remain competitive and be a leading supplier to the tool-making industry, it has to insist on ongoing investment in machinery and CAD/CAM tools. However, the current commercial VDM systems are often expensive because of the involvement of high performance graphics accelerators, multiprocessor graphics workstations and some dexterous immersive devices. Each workstation usually needs dedicated licensed software as well as maintenance supports. Most SMEs cannot afford such high-cost VR systems, and have a clear preference for desktop VR systems over more immersive HMD-based systems (Burdea 1999). With the rapid growth of network technology, the World Wide Web (WWW) has become a primary medium for information sharing and distribution, remote communication and multimedia services via a client-sever approach. In

conjunction with Internetworked 3D graphics techniques such as Virtual Reality Modelling Language (VRML) and the emerging X3D, the new generation of VR applications have begun to be developed via the Internet. Therefore, it is believed that a sharing environment based upon the WWW will allow industries to utilise VR sufficiently and cost-effectively. However, it is observed that little research work had been done in applying Web-based VR technologies to support interactive applications in design and manufacturing for collaborative e-service.

1.3 Research Aims and Objectives

The aim of this research project is to propose a cost-effective approach to applying Web-based VR technologies to support design and manufacturing applications via the Internet. Based on the proposed approach, a distributed VR system as a demonstrator is required to be prototyped. It will allow multiple users such as SMEs to carry out engineering activities in a Networked Virtual Environment (Net-VE) at lower cost, and in less time, as well as facilitating the sharing of design and manufacturing resources through the WWW without extra economic burdens.

In order to achieve these aims, the following objectives were set:

- To investigate the relevant VR and Web-based VR techniques to explore an effective method of Networked Virtual Environments (Net-VEs) by combining VR and network technology.
- To design a cost-effective approach to applying Web-based VR technologies to support interactive applications in the design and manufacturing via the Internet and then to prototype a distributed VR system by using VRML.
- To explore and develop key system components — VR database, VR engine, and network communication in the proposed Web-based VR system.

- To demonstrate the proposed approach through developing a software toolkit — the distributed VRML-based Factory Layout Simulator (VFLS) for factory layout applications on the Web.
- To test and evaluate the software system and distribute the VRML scenes produced by VFLS across the Internet. Furthermore, to integrate with database and HTML-based information system through the Web for information visualization and e-service.

1.4 Contributions

The primary contribution of the PhD research work is the design, development, and implementation of a cost-effective approach to applying Web-based VR to support the design and manufacturing applications via the Internet for collaborative e-service.

Based on the proposed approach, a distributed VR system through the Web has been designed by the author. Its key system components — VR database, VR engine, and network communication have been exploited and developed. It provided the solutions for critical implementation issues including geometric modelling in VR database, simulation engine, and network communication in the proposed Web-based VR system for design and manufacturing applications. Furthermore, a prototype software system — the distributed VR-based Factory Layout Simulator (VFLS) for factory layout applications through the Web has been developed as a key contribution to demonstrate the proposed approach and framework. The major implementation components of the prototype system include: the construction of Virtual Machine Database (VMD); the creation of grid-enhanced 3D coordinate system; the implementation of "*Inline*" mechanism for the creation of complex VR scenes; the composition of 3D transformation for virtual layout operations; and the introduction of the '*Virtual Sensors*' for real-time interactive manipulation in a distributed virtual environment such as WWW.

In addition, the collaboration working experiment has been successfully done by the author over a local or wide area network to validate that the Web-based VR software system — VFLS developed in this project can efficiently support multiple users for collaboration working in design and manufacturing. The final distribution of VRML scenes across the Internet and the integration of database and HTML-based information system have been carried out through the Web. As a demonstrator, the implementation of VFLS has successfully validated the viability of the proposed approach and the usability of the distributed VR system for collaborative e-service.

In summary, this research work has contributed to the research area by developing and applying a Web-based VR approach to support design and manufacturing in order to bridge the gap between VR and distributed applications in the area of design and manufacturing via the Internet. This research work proved that a distributed virtual sharing environment based upon WWW allows industries to utilize VR sufficiently and cost-effectively. The proposed approach and system will enhance and even accelerate the major activities of engineering, including: product concept development and simulation, manufacturing process, optimising assembly lines and workshops design, integrating labour and equipment, etc.

1.5 Thesis Organization

The thesis is organized in seven chapters as below:

Chapter 1: Introduction

This chapter introduces the motivation and challenges in the subject of this research work and presents the main research aims and objectives. It presents the main contributions of the author and the structure of the thesis.

Chapter 2: Background

This chapter describes the background and previous work on VR. It begins by providing an overview of Virtual Reality (VR) technology along with the

component technologies for VR, VR system classification, and VR applications. It then discusses the development of Web-based VR technologies, especially VRML development. Next, the chapter carries out a literature review on the subject of the research work. Evidences show little research work had been done to apply Web-based VR technologies to support interactive applications in design and manufacturing for collaborative e-service.

Chapter 3: Design of a Cost-effective Web-based VR approach in Net-VEs

This chapter firstly presents the concept and features of Net-VEs. Then, it carries out a survey of Net-VE systems and analyses the major challenges. Next, it proposes a cost-effective approach to applying Web-based VR to support design and manufacture via the Internet for collaborative e-service. The distributed VR system using VRML is designed for interactive applications in design and manufacturing in order to benefit multiple users, especially for SMEs through WWW. Finally, it discusses potential applications that can be prototyped based on the above approach.

Chapter 4: Design and Development of the Key Components in the Web-based VR System

This chapter describes the underlying design and development of key components — VR database, VR engine, and network communication in the proposed Web-based VR system to support interactive applications in design and manufacturing. At first, it designs a hybrid modelling method to create 3D model of facilities in VR database. Secondly, it proposes a generic framework of VR engine and the model of interactive simulation. It then presents how the capabilities of interactive simulation in VRML are used to develop interactive dynamic behaviour simulation in the proposed Web-based VR system. Finally, the chapter discusses network communication in the design of Web-based VR systems. It suggests a hybrid method of reliable communication by using multiple protocols for the proposed system in Net-VEs.

Chapter 5: Implementation

This chapter presents a specific implementation of a prototype system for factory layout applications — VRML-based Factory Layout Simulator (VFLS). Furthermore, the chapter presents the software development of VFLS by using the well-known UML approach of object-oriented software engineering. As a demonstrator, the implementation of VFLS has successfully validated the viability of the proposed approach and the usability of the distributed VR system for collaborative e-service.

Chapter 6: Results and Evaluation

This chapter firstly presents results of the VFLS in terms of its functionality and performance of its key components. It examines the contents of VR Database — Virtual Machine Database (VMD) involved in the simulator, and then shows various factory layout scenes produced by this interactive simulator as examples to validate the usability of the software toolkit. It then describes the result of a collaboration working experiment which has shown that the VFLS developed by the author can efficiently support multiple users for collaboration working in design and manufacturing over a local or wide area network. Next, the chapter describes the final distribution of VRML scenes across the Internet and the integration of HTML-based information system through the Web. In addition, the chapter evaluates on the VFLS in terms of its database — VMD and interaction performance. It finally carries out the overall evaluation of a generic Web-based VR system proposed in this research work.

Chapter 7: Conclusions and future work

This chapter summaries the work that has been carried out so far. Furthermore, it discusses the limitations and enchantments in this research. It then presents how this research work carried out by the author in this project can potentially enhance and integrate with the Windchill software solution for advanced Collaborative Product Commerce (CPC). It finally outlines the further work that will need to be done in order to improve and extend the current research work.

Chapter 2

Background

2.1 Introduction

This chapter describes the background and previous research on VR. It begins by providing an overview of Virtual Reality (VR) technology along with the component technologies for VR, VR system classification, and VR applications. It then discusses the development of Web-based VR technologies, especially VRML development. Next, the chapter carries out a literature review on the subject of the research work. The main research projects associated with applying VR to create virtual worlds on the Internet, the previous research work on VR applications in design and manufacturing, and the commercial VM systems such as Deneb/VR have been described. Evidences show little research work had been done to apply Web-based VR technologies to support interactive applications in design and manufacturing via the Internet for collaborative e-service. Finally, this chapter briefly introduces a new emerging Information Technology (IT) market, which is called Collaborative Product Commerce (CPC). CPC enterprise solutions such as PTC's Windchill system are Web-based solutions that use the Internet to allow employees, customers, and suppliers to collaboratively develop, build, and manage products throughout the lifecycle although there is still lack of more effective information representation capabilities which can be dramatically enhanced by integrating with Web-based VR technology.

2.2 VR technology overview

Virtual Reality, also referred to as Virtual Environments (VE), is a novel, developing and exciting area concerned with creating 'representational worlds' with which users may interact through using the interactive 3D computer graphics technology. Virtual Reality research is referred to as a multidisciplinary activity due to its diverse applications. It converges several disciplines such as computer graphics, visualization and simulation, software engineering, hardware design, automatic control engineering, as well as human factors.

Looking back over the historical development of VR, it is interesting to see how each major technological breakthrough brought us one step towards today's VR systems. Since **Ivan Sutherland** first described a computer created illusion: 'The screen is a window through which one sees a virtual world, the challenge is to make that world look real, act real, sound real, feel real.' (Vince 1995). It has taken over 20 years for computer graphics and display technologies to become a cost-effective means of creating virtual reality, and there is still a long way to go before virtual reality becomes cost-effective for all computer users. In the past few years, VR has matured considerably. New hardware platforms and soft environments have been well established and are leading us towards a truly exciting future.

VR is a technology that encompasses a broad spectrum of ideas. The term was defined as a computer generated interactive, three-dimensional environment in which a person can be immersed. There are three key points in this definition. First, this virtual environment is a computer generated three-dimensional scene that requires high performance computer graphics to provide an adequate level of realism. Secondly, the virtual world is interactive, and the user requires real-time response from the system to facilitate interactive with it in an effective manner. Thirdly, the user can be immersed in this virtual environment. The goal of VR is to provide a fully or partially immersive environment.

A VR system provides a domain where a virtual world can be modelled, simulated, visualised and even experienced using the immersive displays (Vince 1995). The computer-generated virtual world provides the domain created by geometric descriptions of objects in the real worlds. With the advent of high performance graphics workstations, it is possible to provide the ability to manipulate 3D models in real time. These 3D models can be observed from any direction; their surface properties can be changed instantly; as well as being able to have different orientations. In other words, a VR system consists of a collection of 3D objects and light sources that are manipulated by animation and physical simulation procedures. It makes users a part of the virtual domain using appropriate devices or soft environments.

The sensation of presence or immersion makes the user feel the virtual domain is real. Using sensors to track the position and orientation of the user, a virtual environment generated by a high-performance computer graphics system can respond to the user's movements. In addition, VR provides real-time interaction for users by using not only a common Graphical User Interface (GUI) but also perceptual and multi-modal interfaces such as gesture, audio and speech recognition.

2.2.1 VR component technologies

In order to provide the sense of presence and the sense of immersion, a typical VR system consists of a high-speed graphics computer that can render at least five million polygons per second, a VR display, a tracking device to sense the user's position, and a device to interact with the virtual environment. These component technologies are briefly described as below.

- ***VR Displays***

  There are two main VR displays technologies: Head Mounted Displays (HMDs) and Projection-Based Displays (PBDs). Since Sutherland's early

research on HMDs in the late 1960s, HMDs have been developed for a range of specific applications, which concerns not only military flight simulation but also the commercial market. HMDs employ a liquid-crystal display (LCD) mounted on a helmet to let the wearer experience a stereoscopic scene. The Cathode Ray Tubes (CRT) Displays usually provide higher resolution but are heavy; LCDs are lighter but have lower resolution. Both types are combined with head tracking for complete immersion in a VR system. Figure 2-1 shows a view of a HMD.

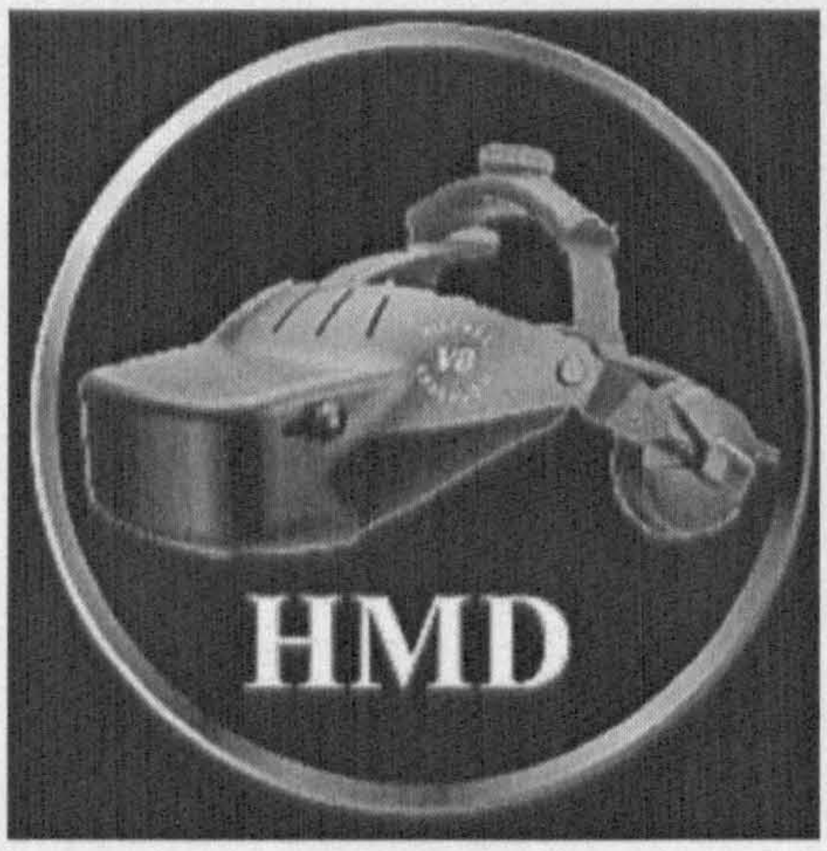


Figure 2-1 A View of a Head Mounted Display (HMD)

Although HMDs provide a useful immersive environment, they can restrict the way virtual environments are presented to a group of people. To overcome such problems, PBDs were developed in the 1990s. PBDs consist of high-resolution projectors (1,024×768 pixels) with large fixed screens set at a distance from the viewer, along with shutter glasses for stereo viewing. PBDs provide high-resolution images in a stable display for multiple users to share. PBD have become the popular VR display technology in recent years. They range from large single-wall configuration to multi-walled immersive rooms. The one wall configuration is often used as a high-resolution display for large audiences. However, such a front-projected screen mounted on the wall could result in the limitation of interaction (Lanzagorta et al. 2000).

Lawrence Rosenblum, et al. (Lanzagorta et al. 2000) at the Naval Research Laboratory (NRL) in Washington developed a "VR Workbench" to display strategic information that could be viewed by a group of users. The system includes a high-resolution projection system, a mirror, and a

table with an imaging surface. A virtual scene, which is created on a high-performance graphics computer, is projected to a mirror and then toward the table's imaging surface. The horizontal surface is suited for tabletop tasks such as medical and terrain visualisation. The Naval Research Laboratory produced the first workbench system in the US in 1994. Since then, many applications have been ported to this virtual reality. Figure 2-2 (a) shows a medical application of the workbench system.

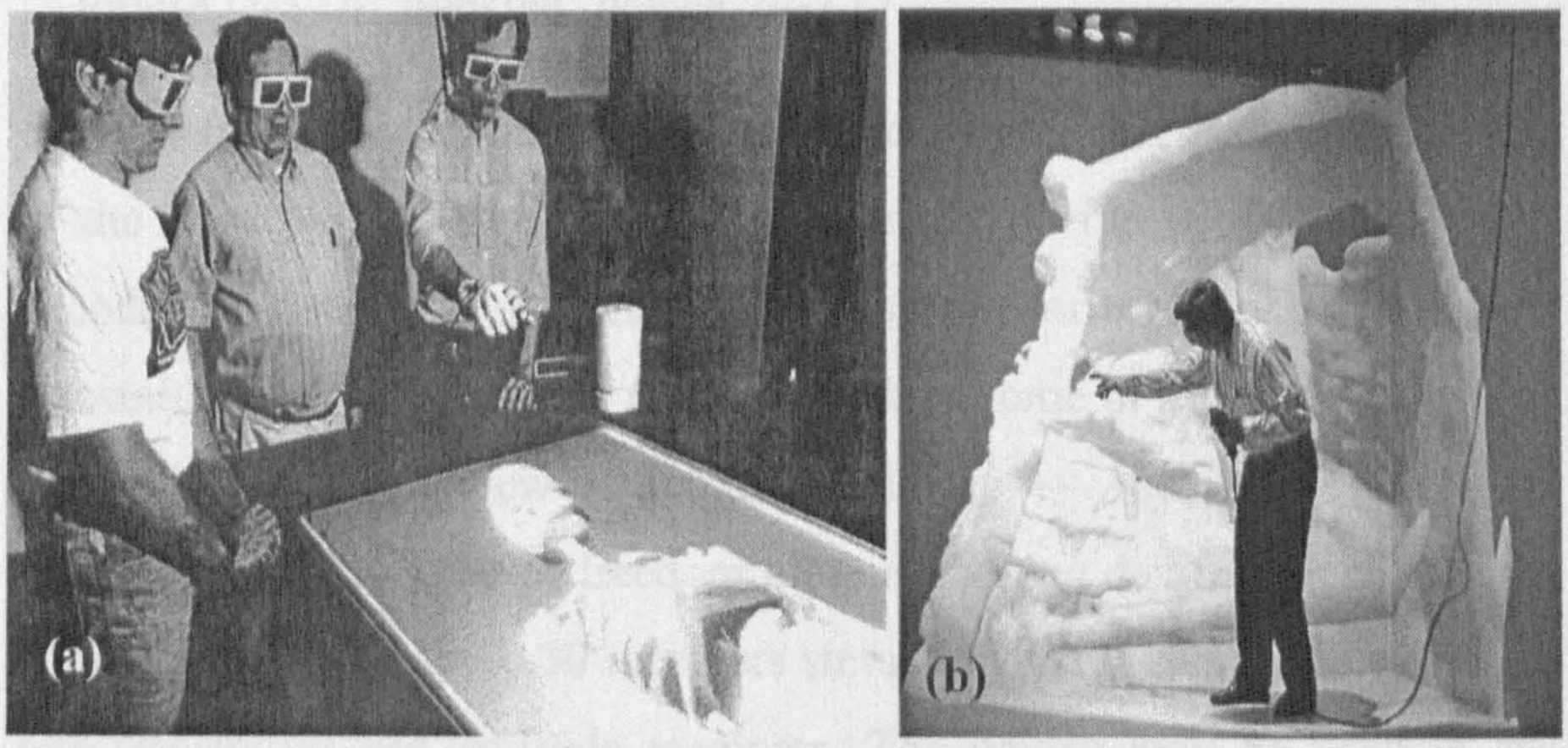


Figure 2-2 (a) The workbench system was applied in a medical application (b) The NRL Grotto—an immersive-room display

Immersive rooms provide surround-view, surround-sound with the multi-walled configuration (Lanzagorta et al. 2000). The system projects stereoscopic images onto multiple walls. The number of walls ranges from three to six. Its size is approximately 10×10×10 feet. Figure 2-2(b) shows an immersive room named Grotto (graphical room for orientation, training, and tactical observation), which is used by NRL for scientific and battlefield visualisation. The image shows a VR application for exploring material microstructure.

- ***Tracking Devices***

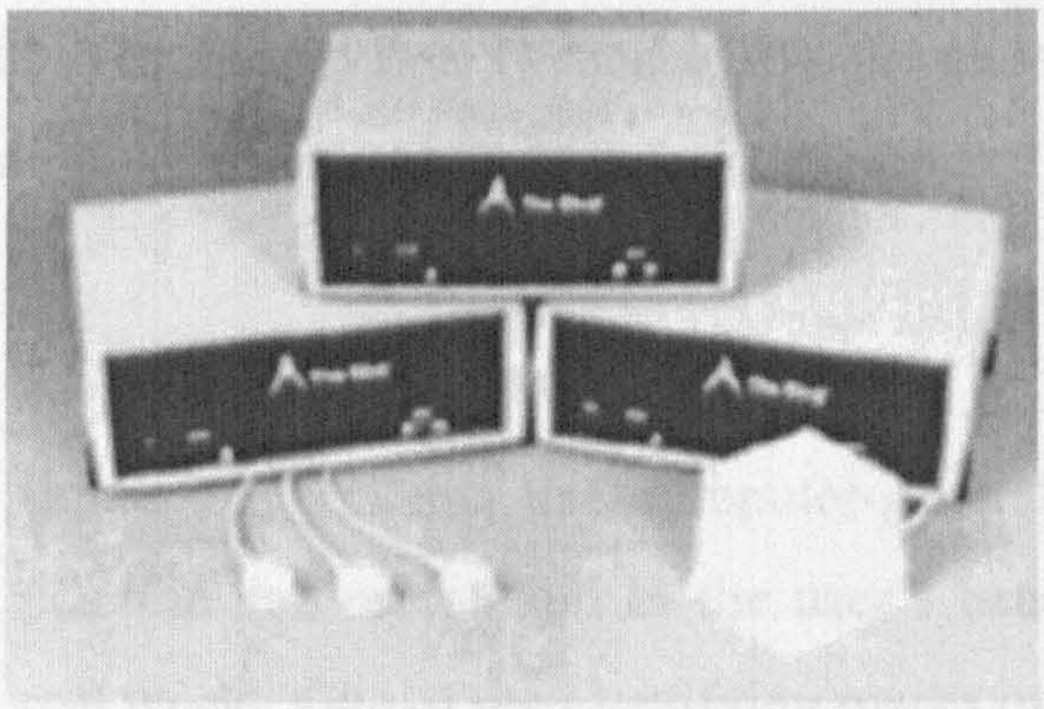


Figure 2-3 A tracking device — Ascension's Flock of Birds

A 3D motion tracker is an essential device to interact with a virtual environment. The most trackers are electromagnetic. They consist of a stationary transmitter and one or more receivers. Typically, a transmitter emits electromagnetic signals that are intercepted by a mobile detector attached to the user. When these signals are received by the detector, they are decoded to determine the relative position and orientation between the transmitter and receiver. The tracking devices come in a variety of forms. Figure 2-3 shows a view of a tracking device — Ascension's Flock of Birds. It is a six-degree of freedom motion tracking device, which can be configured to track up to 30 receivers simultaneously. The unit consists of one transmitter and multiple receivers. The device may be used in an operating range of approximately 10 feet (Vince 1995).

- ***Interaction Devices***

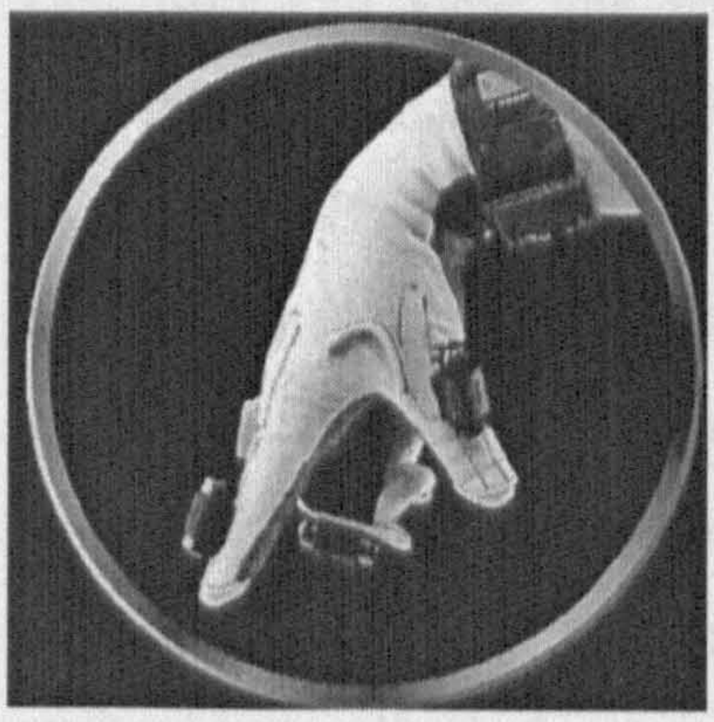


Figure 2-4 A sensor glove—The CyberTouch

A primary interaction device is a sensor glove such as the DataGlove developed by VPL. The DataGlove can monitor the status of the user's fingers, which is achieved with thin fibre-optics attached to the back of the glove's fingers. More complex gloves have ability to measure finger movement and provide haptic feedback such as the CyberTouch developed by Virtual Technologies, Inc. (Vti) (Virtual Technologies Inc. 2001) (see figure 2-4).

2.2.2 VR systems classification

According to **John Vince** (Vince 1995), the classification of VR systems can be divided into three groups: immersive, desktop and semi-immersive, and hybrid.

- ***Immersive systems:***

  Immersive systems use computer-generated scenarios that react to the position and orientation of the user's head instead of user's view of the real world. One of the identifying marks of such systems is the HMD worn by users. These displays block out all the external world and present the wearer with a view that is under the complete control of the computer. The user is completely immersed in an artificial world and becomes separated from the real world. For this immersion to appear realistic, the virtual reality system must accurately sense how the user is moving and determine what effect that will have on the scene being rendered in the HMD.

- ***Desktop and Semi-immersive systems***

  Desktop and semi-immersive systems, on the other hand, leave the user visually aware of the real environments but able to observe the virtual world rather than be fully immersed in it. They traditionally use some display devices such as graphics workstations or large fixed screen PBDs. The users can view and navigate the virtual world just using mouse, keyboards or some appropriate software interfaces such as VRML browser (i.e. Cosmo Player). It also enables to deliver stereo images via special glasses (e.g. shutter glasses) and to navigate by means of a space mouse or space ball. Such systems make it possible to view the virtual world through the window of a conventional screen. Thus they are also called 'through-the-window' systems. Desktop and semi-immersive systems have evolved naturally and appeared simply because computers have become increasingly faster.

- ***Hybrid VR system***

  Hybrid VR system is a combination of the real scene viewed by the user and a virtual scene generated by the computer that augments the scene with additional information. In other words, it generates a composite view for the user. Such systems are also known as 'augmented reality' systems. In Augmented Reality (AR), the user can see the real world around him, with computer graphics superimposed with the real world. Instead of replacing the real world, an AR system supplements it. Ideally, it would seem to the user that the real and virtual objects coexisted. Fundamentally, Augmented Reality is about augmentation of human perception: supplying information not ordinarily detectable by human senses. For example, an AR system can create a three-dimensional model of a fetus scanned inside a womb with an ultrasonic sensor, then overlay it on top of the mother's womb. The goal is to give the doctor "X-ray vision" enabling him to "see inside" the womb. At present, Augmented Reality is a growing group in virtual reality research. The potential of Augmented Reality is great. Basically, applications of this technology employ virtual objects to aid the user to understand his/her environment. Researchers working with Augmented Reality systems have proposed them as solutions in many domains. The areas that have been discussed range from entertainment to medical training and manufacturing process simulation (Caudell and Mizell 1992).

The obvious difference among the above three categories of VR systems is the immersiveness of their system. An immersive system strives for a totally immersive environment. In contrast, a desktop and semi-immersive system is highlighting the sense of presence rather than fully engagement. A hybrid system is augmenting the real world scene necessitating that the user maintains the sense of presence in that world. It provides a mechanism to merge the virtual image with the real view to create the augmented display.

2.2.3 VR technology applications

At present, more and more applications fall into VR. VR definitely provides a true three-dimensional interface to these applications. Particularly, it not only provides a powerful technology to explore familiar environments associated with planes, vehicles, cities, schools, shops and hospitals, but also to explore unfamiliar environments found in molecules, atoms, galaxies, viruses, bacteria and crystals (Vince 1995). For instance, a tourist can know how to travel in a strange city and know what it looks like using a city simulator; a surgeon can learn new surgical skills using a medical simulator. The main applications are described below.

Virtual Manufacturing

Virtual Manufacturing (VM) takes advantage of Virtual Reality (VR) technology to provide an e-manufacturing solution in a 3D visual and interactive way. It has shown the powerful potential of VR by being combined with the latest generations of CAD/CAM tools to aid product development. For example, time and money can be saved using virtual prototyping. A CAD design can be exported to a VR software application to create a virtual prototype for evaluation. The prototype can be further used throughout the lifecycle of product development to incorporate changes or new ideas, or even get client approval. VM has made VR no longer state-of-the art but rather an innovation technology to support modern industry. Engineers have applied VM in a wide range of manufacturing applications: product design and prototyping, factory layout design and visualisation, assembly process planning and simulations, machinist training, as well as many other applications. More details are provided in Section 2.5.

Scientific Visualisation

Scientific visualisation takes advantage of VR technology to display scientific data in a 3D way to enhance the understanding and analysis of the phenomena, or reveal the hidden problem behind the data. As complex multidimensional data sets are usually involved in these applications, data analysis and exploration can greatly benefit from VR's 3D interface and further enhance the process of

scientific understanding. For example, astronomers use VR to visualise the universe and move around the data they have collected. They have already made startling discoveries using VR, including the way in which huge numbers of galaxies tend to clump together in threads, ribbons, and clusters with vast distances between them.

Information Visualisation

Information visualisation becomes widespread rapidly because Hypertext Markup Language (HTML) led to a population explosion on the Web by implementing a graphical interface. Visualisation and graphics will be the dominant technologies to show the information to people, or guide people to the information. VR technology provides a powerful solution for information visualisation on the Internet. For example, prospective students are allowed to "walk" through a virtual campus delivered across the Internet. Such applications can help the potential students to choose their favourite universities efficiently. On the other hand, it is an excellent means of promoting a university to a much wider audience.

2.3 The development of Web-based VR technologies

Although networking and computer graphics are regarded as two different disciplines, these fields begin to converge in order to support collaborative exploration and visualisation on the Web. The rapid growth of network technology provides new opportunities for VR applications. The Web-based VR technologies such as Virtual Reality Modelling Language (VRML) have been developed to support the interactive applications on the Internet (Rhyne et al. 1997) (Dieter and Jucknath 1996). This section describes the development of Web-based VR technologies. It presents components of Web-based VR, an overview of VRML, the evolution of VRML/X3D, the design goals and features of VRML, how VRML fits into the WWW, and descriptions of the future development of VRML.

2.3.1 Aspects of Web-based VR

There are six aspects involved in a Web-based VR system, including connectivity, content, interaction, economics, applications and personal impacts (Brutzman 1997).

1. **Connectivity** refers to the capacity, bandwidth, and protocols on the networking infrastructure. A networking protocol is a set of rules or procedures for governing data exchange on the Internet.
2. **Content** includes any type of information and data sets that are transported via a networking protocol. VRML technology and MPEG-4 technology can provide 3D graphics content and mixed media scenes on the Web.
3. **Interaction** involves a sense of presence, and the ability to both access and modify content.
4. **Economics** refers to the expense of access and use of connectivity, content and interaction as well as methods of handling Web transactions.
5. **Applications** determine requirements for Web-based VR. With the expanse of the Web, the demand to provide cost-effective VR applications for distributed multiple-user increases. They often include 3D games, interactive instruction, scientific experiments, and distributed simulations in a Net-VE.
6. **Personal Impacts** refer to the sense of wonder associated with Web-based information discovery and exploration.

2.3.2 Virtual Reality Modelling Language (VRML)

Nowadays, VRML is the key technique in the area of Web-based VR. It has become an International standard (ISO/IEC 14772) for describing interactive three-dimensional scenes delivered across the Internet/Intranet. VRML adds the next level of interaction, structured and extra dimensions (z and time) to the online experience (Pesce 1995).

VRML is designed to be used on the Internet, Intranet, and local client systems. It is also intended to be a universal interchange format for integrated 3D graphics and multimedia. In the past few years, more and more applications such as information visualization, architecture and education require intensive interaction, user participation and exploration. These applications concern how information can be experienced in 3D visual and interactive manner. Such requirements cannot be achieved by HTML based on text and image. VRML defines the commonly used semantics applied in 3D interactive applications such as hierarchical transformations, light sources, navigation information, viewpoints, geometry, animation, collision detection, appearance properties, and level of detail (LOD). VRML provides the technology that combines three dimensions, two dimensions, text, and multimedia into a coherent entity. As this entity cooperates with scripting languages and Internet capabilities, an entirely new style of interactive applications becomes possible.

Therefore, VRML is an effective technique and a good approach to create a Web-based VR system on the Internet (Seiler and Schafer 1998) (Yoshiaki 1998) (Rolland et. al. 2002). It makes VR systems more popular and closer to our life. In this way, it is not difficult for the common audiences to get the benefits of VR technology. At this point, VRML, as a simple, multiplatform language, provides a solution to generate and publish VR applications on Web pages. It definitely plays an important role in the on-line virtual communities as the foundation for cyberspace.

2.3.3 Design goals and features of VRML2.0

VRML 2.0 was designed and created by adding some features that are able to interact and sense with virtual environments. One of the primary goals in designing VRML was to ensure that it at least succeeded as an effective 3D file interchange format. The specific design goals for VRML 2.0 are described as follows:

- **Simplicity**

  Just as the simplicity of HTML led to the explosive growth of the WWW, the simplicity of VRML 2.0 make it possible to create the virtual world with any commonly used text editors and a wide variety of VRML 2.0 browsers.

- **Optimisation**

  VRML 2.0 is designed to be highly optimised. The optimisation of VRML 2.0 makes the creation and implementation of the virtual world as easily as possible.

- **An interchange format**

  In order to make VRML 2.0 be commonly accepted as a file interchange format, VRML 2.0 files must be read and written as easily as possible for applications. One of the critical design decisions was whether it would be best to add animation and interaction capabilities to VRML by making VRML a full-fledged programming language. Programming languages are very powerful, but they make poor file interchange formats.

The needs of both VRML browsers and VRML editors are considered in the VRML 2.0 design. One property of a good file interchange format is composability. It should be capable with compose files created by different users or tools together into a new one. Another property of VRML is to cut and paste objects between VRML worlds easily just as HTML does between HTML documents.

- **Scalability**

  The fundamental design goal for VRML 2.0 should be able to match the increased larger worlds and faster machines and networks. VRML is designed to scale in three ways as follows.

1. It should be potential possible for a VRML browser to handle a world distributed across the Internet that contains millions or billions of objects.
2. VRML should work well with using both very powerful and very inexpensive machines, and scale well with increased hardware performance.
3. VRML worlds should fit into the scalability of network performance, from the commonly used 14.4K modems to high speed network connections in the future.

The major features of VRML are listed as follows.

- **Scene Graph Structure**

  VRML files describe 3D objects and worlds using a hierarchical scene graph. Entities in the scene graph are called nodes. VRML 2.0 defines 54 different node types shown in Table 2-1, including various types of grouping, geometry primitives, properties, event and behaviour, system and miscellaneous nodes such as sound and fog etc. Different types of data from an integer number to a 3D array are stored in 20 different types of fields of these nodes.

  The VRML nodes can contain other nodes and even can be contained in more than one node, but a node must not contain itself. This hierarchy of nodes is called the scene graph. In this way, VRML files exist in a style of parent-children hierarchical structure. Such a structure makes it easy to create large worlds or complicated objects by inheritance from other subparts.

  In VRML files, each grouping node defines a coordinate space as the local coordinate system for its children. This coordinate space is relative to the coordinate space of the node of which the group node is a child. Such a

node is called a parent node. This means that transformations accumulate down the scene graph hierarchy.

A VRML file contains zero or more root nodes. The coordinate system in which the root nodes are displayed is called the world coordinate system.

System and Miscellaneous Nodes	Group Nodes	Object Nodes	Property Nodes	Event Nodes
WorldInfo	Anchor	Box	Appearance	Script
NavigationInfo	Billboard	Cone	Color	PlaneSensor
Viewpoint	Collision	Cylinder	Coordinate	CylinderSensor
Background	Group	Sphere	Normal	SphereSensor
DirectionalLight	Inline	ElevationGrid	FontStyle	ProximitySensor
PointLight	LOD	Extrusion	Material	TimeSensor
SpotLight	Shape	PointSet	ImageTexture	VisibiltySensor
Sound	Switch	IndexedLineSet	MovieTexture	TouchSensor
AudioClip	Transform	IndexedFaceSet	PixelTexture	CoordinateInterpolator
Fog		Text	TextureCoordinate	NormalInterpolator
			TextureTansform	OrientationInterpolator
				PositionInterpolator
				ScalarInterpolator
				ColorInterpolator

Table 2-1 VRML2.0 nodes

- **Event and routing mechanism**

  An event or routing mechanism defined by VRML2.0 allows nodes in the scene graph to communicate with each other or pass messages from generators to receivers. Such mechanism plays an important role in response to environmental changes or user interaction.

  VRML 2.0 contains nodes that handle the event generation and routing mechanism, as well as nodes that participate in the audio-visual

representation of objects in the scene graph. Script nodes allow the world creator to define behaviours arbitrarily by binding any supported scripting language such as Java and JavaScript (Carson et al. 1999). A Script node is able to execute a script function for processing the relevant event. Script nodes also allow events to be processed based on the order of user actions. Such scripts can dynamically add or delete routes and thereby change the event-routing topology. These script nodes can be inserted between event generators (typically sensor nodes) and event receivers.

Additionally, Interpolator nodes defined by VRML are mainly used to handle built-in behaviour by performing simple animation calculations. They are often combined with a TimeSensor to make objects move.

- **Sensors**

  Sensors defined by VRML are the basic user interaction and animation primitives. They have the ability to sense and detect changes over time, the state of input devices or the position of objects. Sensors only generate events; they act on the scene in terms of being combined with routing mechanism.

There are different types of Nodes of Sensors in VRML 2.0 including:

1. TimeSensor node: It detects changes of time and generates events as time passes. TimeSensor nodes can be used for driving continuous simulations and animation, controlling periodic activities and behaving as an alarm clock.
2. CylinderSensor, PlaneSensor, SphereSensor, TouchSencor nodes: They generate events when the pointing device is activated while the pointer is indicating any children geometry nodes of the sensor's parent group. They are the basis for all user interactions through some input devices. They are classed as pointing-device sensors.

3. ProximitySensor, VisibilitySensor nodes: They detect changes related to the motion of the viewer or objects in the virtual world. They generate events when the viewer enters, exits, and moves within a region in space. They are often used for activating or deactivating some behaviour or animation. They are classed as environmental sensors.

   With the development of VRML, it is possible to create more new types of sensors for supporting various types of input devices such as speech channels and so on.

- **Prototyping Mechanism**

  A prototyping mechanism defined in VRML 2.0 allows geometry, properties, even behaviours and animation to be encapsulated and reused. Such mechanism makes it possible for VRML to define a new node type by combining with an existing node types. On the other hand, it also can make VRML easier to use and can reduce the size of VRML files.

- **Distributed Scenes**

  There are two primitives defined by VRML2.0 that have the ability to create a single VRML world based on its subparts that are distributed across the WWW. They are described below:

  1. Inline node: It enables to include another VRML file stored anywhere on the WWW.
  2. EXTERNPROTO statement: It enables to fetch new node definitions from anywhere on the WWW. More generally, EXTERNPROTO allows nodes to be defined external to the VRML file and it is the basic extensibility mechanism for VRML.

2.3.4 Accessing VR through VRML browser

A VRML world can be accessed, viewed and navigated by using appropriate software interfaces — the VRML browser. In other words, a VRML browser provides the user a key to open the door to Virtual Reality on the Web. This section provides an introduction to VRML browsers.

A VRML browser provides a mechanism that has the ability to interpret, execute and present VRML files to the user. Therefore, one of its major tasks is to display objects and sounds in the scene graph by presenting the transformation hierarchy to the user. The transformation hierarchy describes subparts of the virtual world. It is by a VRML browser that the user feels the sense of presence from navigating a Web-based VR application. A virtual world can be presented by a browser; the world can then be navigated in the browser. A user also can view it from a particular location known as viewpoint. The viewpoint can be moved through the virtual world by the navigation paradigms defined by the browser such as walking or flying. In addition to navigation, the browser should be responsible for the interaction between the virtual world and the user. It may provide a mechanism allowing the user to interact with the world through sensor nodes in the scene graph hierarchy. Sensors respond to user interaction with geometric objects in the world, the movement of the user through the world, or the passage of time.

Figure 2-5 shows a conceptual model of a VRML browser (Carson et al. 1999). The browser is illustrated as a presentation application interface that accepts user input in the forms of file selection and user interface gestures using an input device. The browser has three main components: Parser, Scene Graph, and Audio/Visual Presentation.

- The Parser component reads the VRML file and creates a Scene Graph.
- The Scene Graph component consists of a Transform Hierarchy with the nodes and a ROUTE Graph that forms the connections between nodes.
- The Audio/Visual Presentation component displays the graphics and audio rendering of the Transform Hierarchy that feeds back to the user.

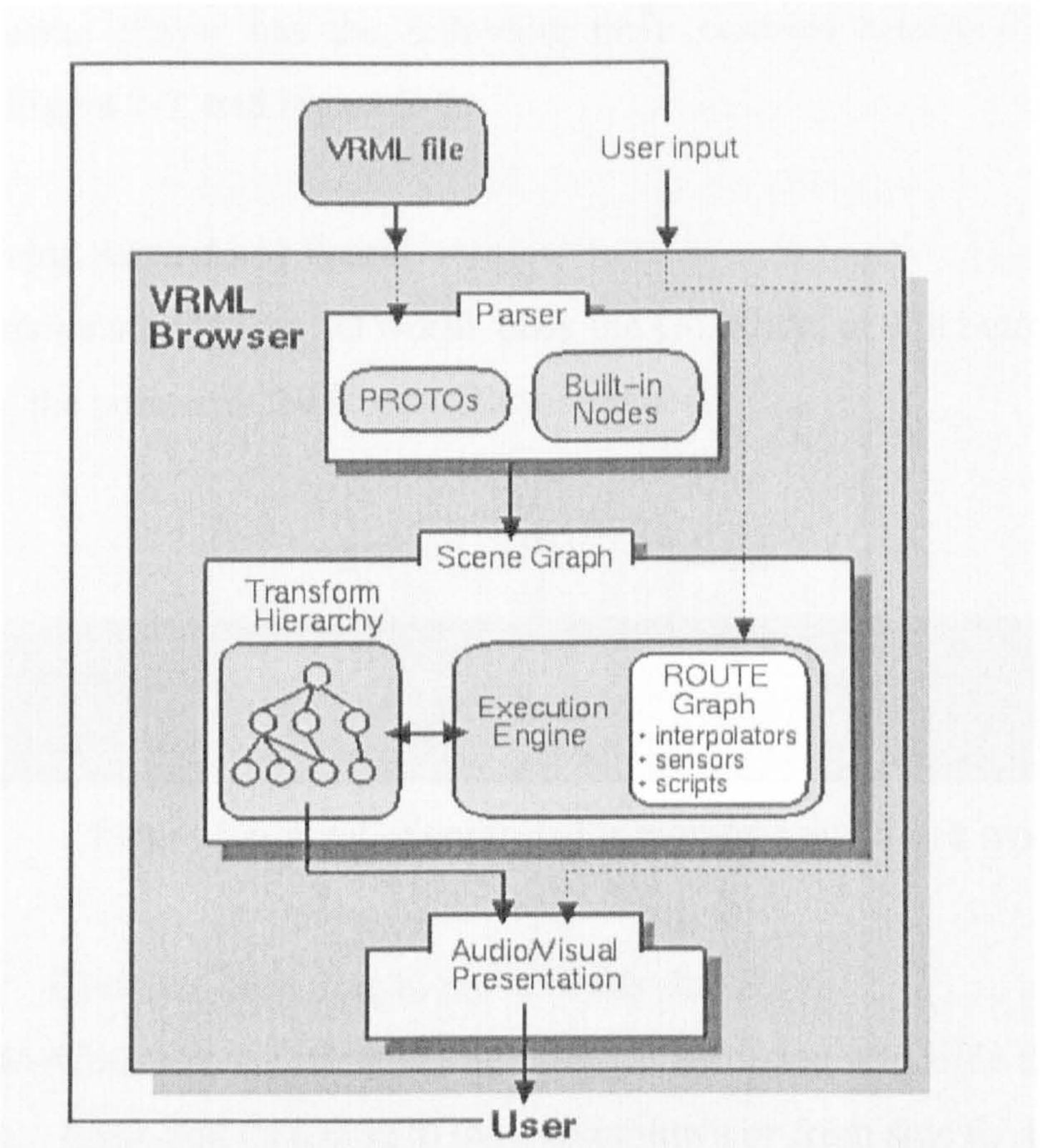


Figure 2-5 Conceptual model of a VRML browser (Carson 1999)

Additionally, an Execution Engine is involved to handle events, generate the ROUTE Graph, and make changes to the Transform Hierarchy in the Scene Graph. Generally speaking, the ROUTE Graph component (sensors) and the Audio/Visual Presentation component (navigation) respond to User input. The Audio/Visual Presentation component carries the feedback to User input. It is such a cycle that exists in a VR system to perform the interaction between users and virtual worlds through a VRML browser.

At present, there are more and more VRML browsers such as Cosmo Player, WorldView, CASUS Presenter, RealSpace Viewer and so on. The most popular one is Cosmo Player. Cosmo Player as a plugs-in of the Web browser allows the Webpage reader to access and explore 3D worlds generated by using VRML. These 3D worlds often integrate with other types of multimedia, like sound and

movies. Cosmo Player has the following main controls briefly illustrated in Figure2-6, Figure 2-7, and Figure 2-8:

1. Moving Around in a World

   To move around in a 3D world, click the Go, Slide, or Tilt button and then drag the pointer in the Cosmo Player window.

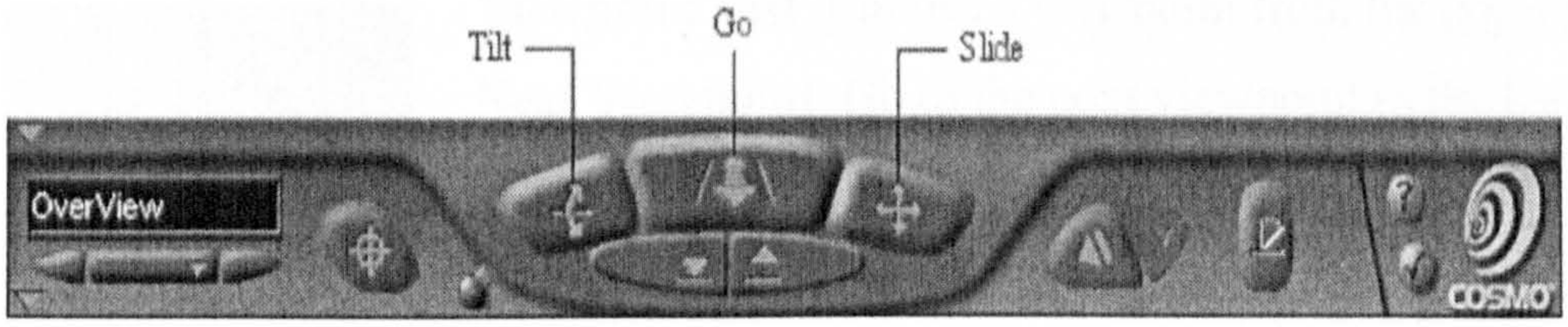


Figure 2-6 A set of controls for moving around in a world

Go Click and then drag to move in any direction.

Slide Click and then drag to slide straight up down or to slide right or left.

Tilt Click and then drag to look up or down or from side to side.

2. Examining Objects

   To examine objects in a 3D world, click the Rotate, Pan, or Zoom button and then drag the pointer in the Cosmo Player window.

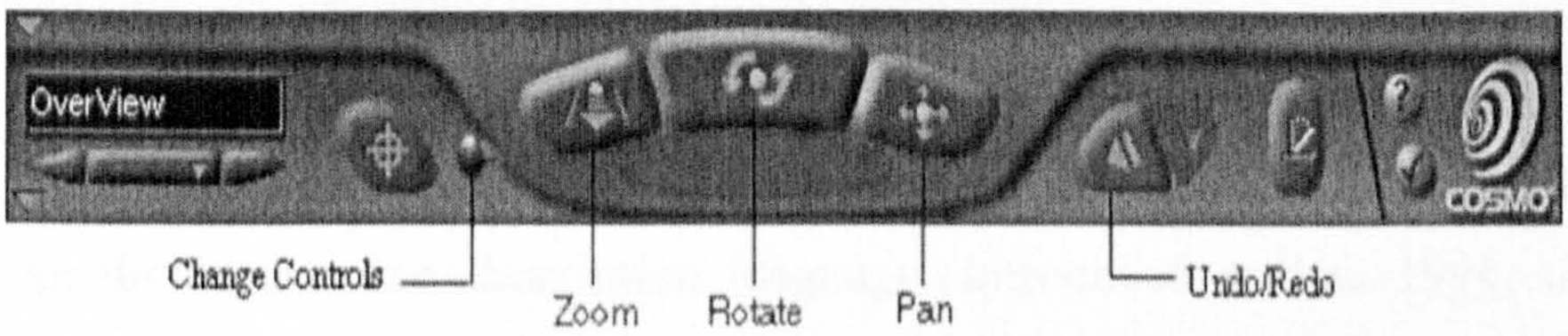


Figure 2-7 A set of controls for examining objects in a world

Rotate Click and then drag to rotate an object.

Pan Click and then drag to pan right, left, up, or down.

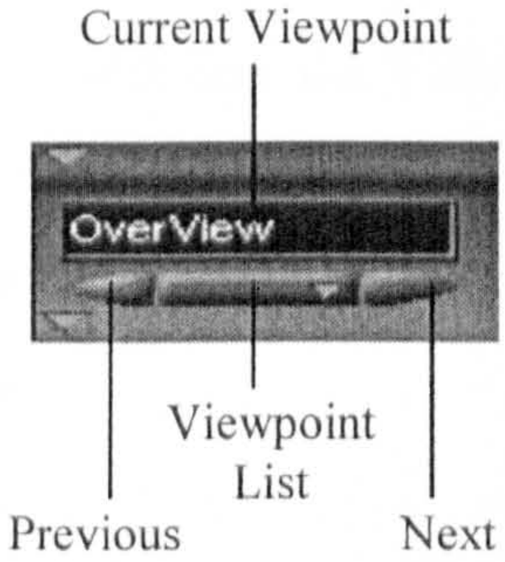
Zoom Click and then drag up to zoom in or drag down to zoom out.

Change Controls Click and switch from one set of controls to another

Undo/Redo Click and then retrace users' steps.

3. Choosing Viewpoints--another way of moving through a world

Authors of 3D worlds can set viewpoints--places of interest--for you to visit. You can move from one viewpoint to another by choosing from the Viewpoint list or clicking the Next Viewpoint or Previous Viewpoint button.



Viewpoint List Choose a viewpoint from the pop-up list.

Next Viewpoint Go to the next viewpoint in the list.

Previous Viewpoint Go to the previous viewpoint.

Current Viewpoint Show the current viewpoint.

Figure2-8 Choosing viewpoints

2.3.5 The evolution of VRML/X3D technology

The Virtual Reality Modelling Language (VRML) has become a tool for building interactive 3D content and applications on the Internet. It has made VR on the Web accessible to users in a wide range of applications. The emerging Extensible 3D (X3D) standard can also be called the next-generation VRML. This section briefly describes the evolution of VRML/X3D technology.

VRML is an open specification originally based on the Silicon Graphics Inc. (SGI) OpenInventor scene description language. Introduced in late 1994, the VRML 1.0 standard provided an object-oriented construct augmented by hypermedia links. VRML 1.0 allowed for scene generation by Web browsers on various computing platforms, including PC computers as well as UNIX workstations. The interaction model of 3D VRML browsers is a client-server approach, similar to most other Web browsers. 3D browsers are usually embedded into 2D browsers (e.g. Netscape or Microsoft Internet Explorer). A VRML source file is downloaded from the Internet server to the Web client for exploring the 3D VRML world. The user interacts with this 3D world on the client computer.

The VRML 2.0 standard was released in 1996. It addressed real time animation issues on the Web. VRML 2.0 provided local and remote hooks to graphical scene description. Dynamic scene changes are simulated by any combination of scripted actions, message passing, user commands, or behaviour protocols such as Java. VRML 2.0 (VRML97) was approved as the International Standard (ISO/IEC 14772) file format for describing interactive 3D multimedia on the Internet in 1997. In addition, the VRML External Authoring Interface (VRML-EAI) as a new development has been defined to become a standard programmer interface, which allows technical users to write external programs that communicate with a VRML browser.

As the Extensible Markup Language (XML), MPEG-4, and additional 3D Web technologies began to emerge, efforts are being made to develop the next-generation VRML (VRML-NG) such as X3D. X3D will be 3D standard for the Web by combining VRML97 with XML. Its specification proposal is ready for submission as a potential ISO standard.

2.3.6 Summary

This section provided an overview of Virtual Reality (VR) technology along with the component technologies, VR system classification, and VR applications. It then discussed the development of Web-based VR technologies. Computer graphics technology and networking technology have begun to converge in order to support collaborative applications which information and data can be experienced in 3D visual and interactive manner on the Web. The rapid growth of network technology provides new opportunities for low-cost VR applications distributed via the Internet. Meanwhile, VRML as an International standard (ISO/IEC 14772) of Web-based VR for describing interactive three-dimensional scenes delivered across the Internet/Intranet has become an effective technique to create Web-based VR systems. It will make common audiences get the benefits of the state of the art VR technology in a sharing environment through the Internet.

2.4 A literature review of VR on the Internet

Generally, most VR applications require high bandwidths as supported by fibre optics and digital broadcasting. However, with the rapid growth of the Internet, the World Wide Web (WWW) has become a primary medium for exchanging information and services. Thus, the Web will provide a sharing environment for the next generation of VR applications. Networking and VR have to converge in order to support distributed interactive applications for collaborative working and multiple-user access. Recent research has shown the powerful potential of such systems. This section examines the latest developments of VR on the Internet by reviewing the previous research and related projects, including: commercial applications, online education and entertainment, information visualization, data mining, collaborative working, and medical training.

2.4.1 Commercial applications

In a relatively short period of time, the Internet has become an important medium for commercial applications. Some researchers have begun to work on 3D product presentation in order to deliver the virtual design exhibition online (Dauner et. al. 1998) and Internet-based collaborative product design with assembly features (Shyamsundar and Gadh 2001). **Klaus Bauer** (Bauer 1998) in the Computer Graphics Centre at Darmstadt investigated and reported on the latest research on the problems of modelling virtual supermarket. He describes the advantages of 3D environments for shopping applications on the Internet and describes the prototype of a software tool that automatically generates 3D models of virtual supermarkets. These supermarkets are distributed on the Internet. The main feature of the system is that it facilitates the generation of individually customised supermarkets for each customer's specific needs and demands.

Nick Burton, Alistair Kilgour, and **Hamish Taylor** (Burton 1998) at Heriot-Watt University study the problems of marketing a product on the Internet. They

investigated the possible use of VRML 2.0 to enhance the Web-based marketing of electronic bagpipes. The case study raised many issues such as modelling, animation, interaction, and sound. They also evaluated the strengths and weaknesses of VRML 2.0 for product marketing. The strengths include: Web based; 3-D; relatively easy to create; demonstration of product behaviour. The weaknesses include: long download time; slow rendering of complex scenes; low-level modelling and behaviour animation capability.

2.4.2 On-line education and entertainment

In this millennium, museums face various challenges. Some people view a visit to a museum as an educational experience, while others see it as a leisure experience. **William Mitchell** (Mitchell 1998) at Manchester Metropolitan University describes how to apply VR technology to meet these challenges. They developed two projects: the first was based upon the tomb of the Egyptian noble Menna, and the second was a virtual environment representation of the pyramid builders' town of Kahun. Their research focused on how the virtual models were created and evaluated how such virtual worlds were explored by different groups of users. The evaluation uncovered several general interface and interaction issues. The findings pointed to a wider problem with designing interfaces for Internet-based applications. The actual form of the interface is dependent on the user and the particular platform and browser software.

Another interesting theme of online education is the 3D reconstruction of the famous history heritages and sites such as Ironbridge, London Bridge etc. They have been represented by VRML-based virtual worlds and delivered across the Internet on **British Broadcasting Corporation** (BBC) online (BBC online 2001) as shown in Figure 2-9. The Ironbridge VRML world was created as a guide to how the world's first iron bridge was built on the banks of river Severn after centuries of stone and wood. The London Bridge VRML world represented how the true London Bridge looked like in its prime in the 1500s. However, the models in the virtual worlds would need to be improved in terms of realism.

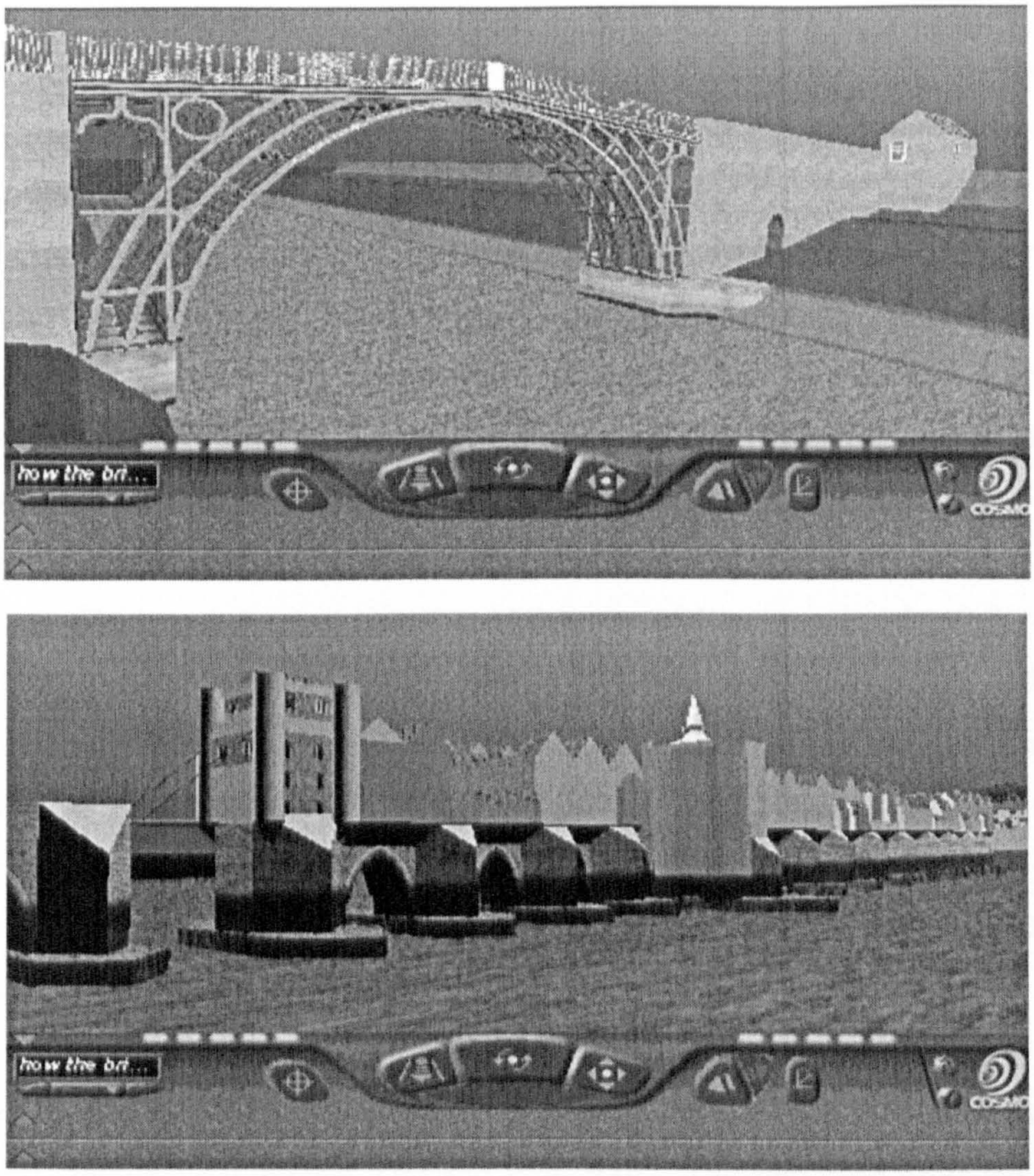


Figure 2-9 The 3D reconstruction of IronBridge and London Bridge

In addition, **Stephen N. Matsuba and Bernie Roehl** brought virtual theatre to the Web (Matsuba and Roehl 1999). This project required a mixture of 3D graphics, efficient networking, and strong content. They discussed a production of "A Midsummer Night's Dream," which was presented over the Internet in real time using VRML and Java. It attempted to create a new medium using VRML on the Internet and also examined the technology needed to create live, streaming, real-time 3D animation for the Internet as well as some of the challenges that lie ahead. Particularly, the streaming audio and motion data involved should be improved.

2.4.3 Information visualisation

Visualisation and graphics are the dominant technologies used to show information to people or guide them to information. Researchers have adopted VRML for cost-effective information visualisation while creating shared virtual environments. VRML worlds let users search and discover information from a representational 3D virtual world through the Internet. Users can view and navigate it on the Web at any time or any place. VRML has been used to create a university campus or a city for interactive student orientation, architectural reviews, and real-time walkthroughs from a distance.

Christine Clark and **Adrian Clark** at University of Essex explored a different use of VRML in order to provide interface to information systems (Clark 1996). Three examples are examined: a model of the University of Essex campus, which interfaces to the homepage of departments on the Web; a model of a theatre, which interfaces to a booking system; and a developing VRML interface to the classic *Adventure* game. However, the architectural models in the systems lacked of realism and the navigation speed was very slow.

Li Jin (the Author) designed an effective approach to add environmental aspects to VRML architectural models* (Jin and Wen 2001). This approach provided the VRML worlds with greater realism and better visual quality while maintaining an acceptable navigation speed. This approach has been applied to adorn the virtual campus of Hull University with other general virtual worlds such as a virtual castle. In this project, a hybrid modelling method was proposed to model complex organic objects such as trees, flowers and so forth. The virtual campus as shown in Figure 2-10 has been delivered across the Internet.

---

* This research work has been published in **the Journal of IEEE Computer Graphics and Applications, vol. 21, no. 1, Jan/Feb 2001**

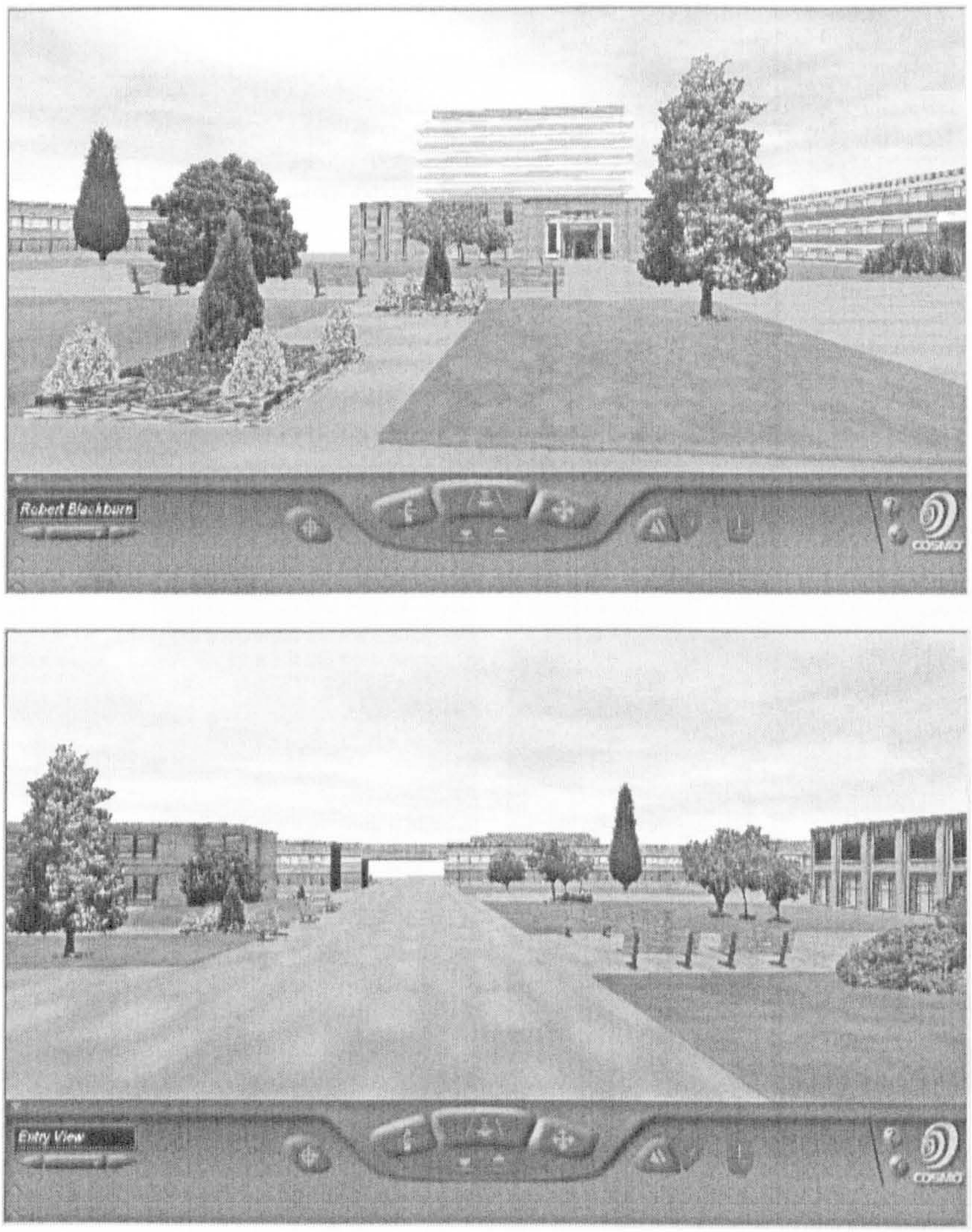


Figure 2-10 The views of the virtual campus of Hull University on the Web (developed by the author)

Similarly, tourists can familiarise themselves with a foreign city by taking a virtual tour of the city on the Web. Meanwhile, to take advantage of the local tourism industry, the local government can construct VR worlds of the resorts and holiday sites, which interfaces to a booking system, to attract more tourists. **The Planet Company** created the first virtual city on the Web (Planet com. 2001). Then, a virtual city was developed as part of a "Human Exposure in Urban Environments" project (Huber 2001) for the United States Environmental Protection Agency (US EPA).

Gareth Ennis and Malcolm Lindsay at University of Strathclyde, Glasgow, discussed how the original 3D Glasgow City model was represented and constructed. They designed an application — The Glasgow Directory for use over the Internet. This system provided users with the ability to search for information about the city or participate in a virtual tour by using VRML (Ennis and Lindsay 2000). Figure 2-11 shows the views of virtual Glasgow modelling-representing in VRML.

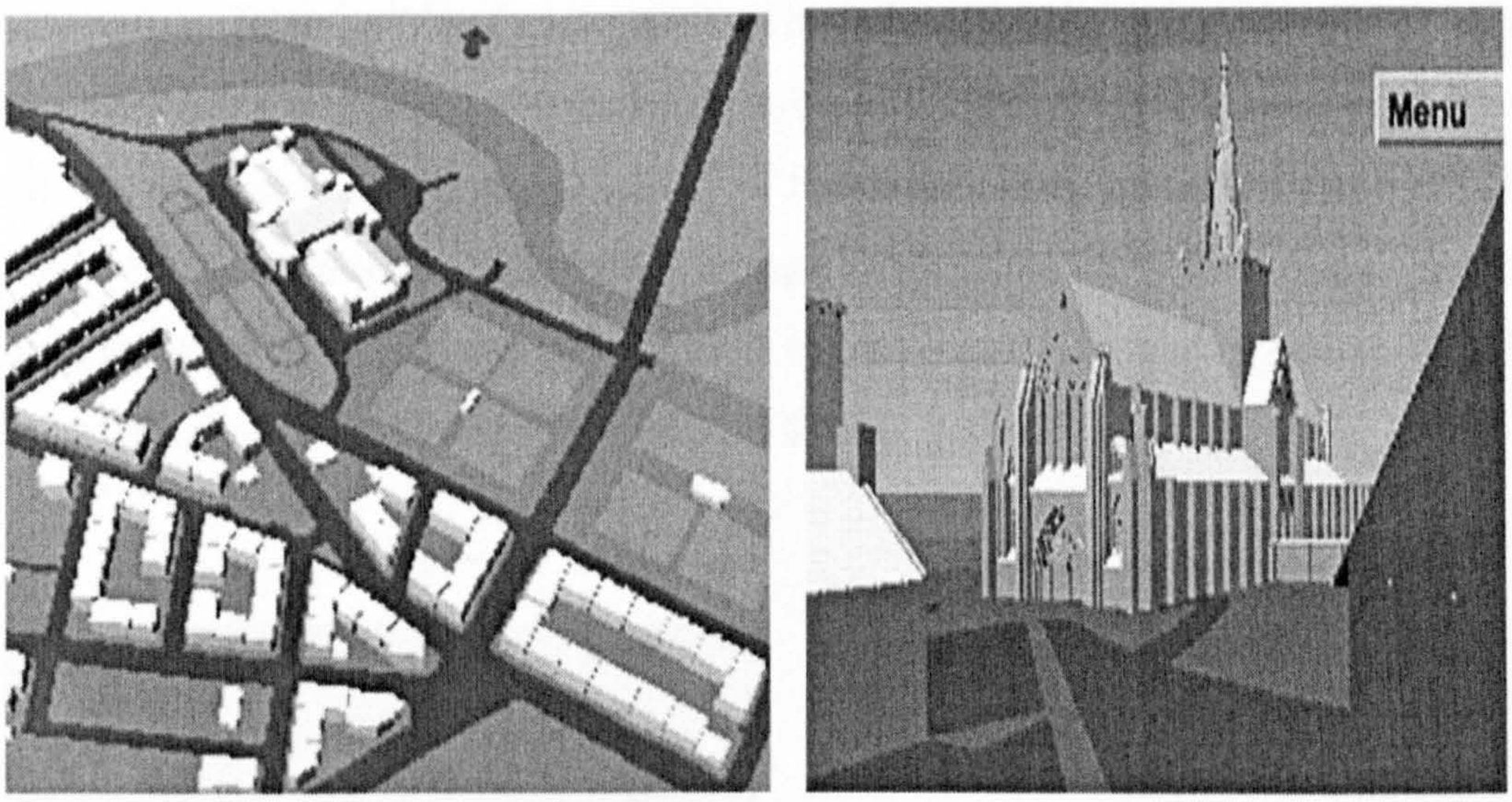


Figure 2-11 The views of virtual Glasgow modelling-representing in VRML

2.4.4 Data mining

Information drilling and interactive data querying are sometime also referred to as visual data mining and will become a major activity on the Web. **Mikael Jern** explained how HTML's Image Map and VRML's anchor code and multiple predefined viewpoints were used for information drilling (Jern 1998). He pointed out that 3D visualisation technology closely integrated with the data warehouse and multidimensional abstract and geospatial models over the next couple of years. It is now possible to create desktop visualisation applications that let users interact with database with larger data sets over the network using both 2D and

3D interaction modes. However, the future improvements in information visualization on the Web would include: semi-immersive VR navigation using visual user interface technology; real-time visualisation of very large datasets; collaborative multi-user visual environment.

Traditionally, mine plans and sections in 2D demand to represent 3D information. **Keith Russ and Andrew Wetherelt** at University of Exeter proposed a new interactive method of data visualisation by using VRML to model the related information (Russ and Wetherelt 1999). They chose a local mine as a case study, due mainly to the amount of coordinate information available and also because no one has created a complete 3D model of the mine. They presented the methods employed in data collection and editing, conversion to VRML, and model construction. Figure 2-12 shows a typical tunnel outline created by using VRML along with the general view of the 400fm, 420fm, and 445fm levels.

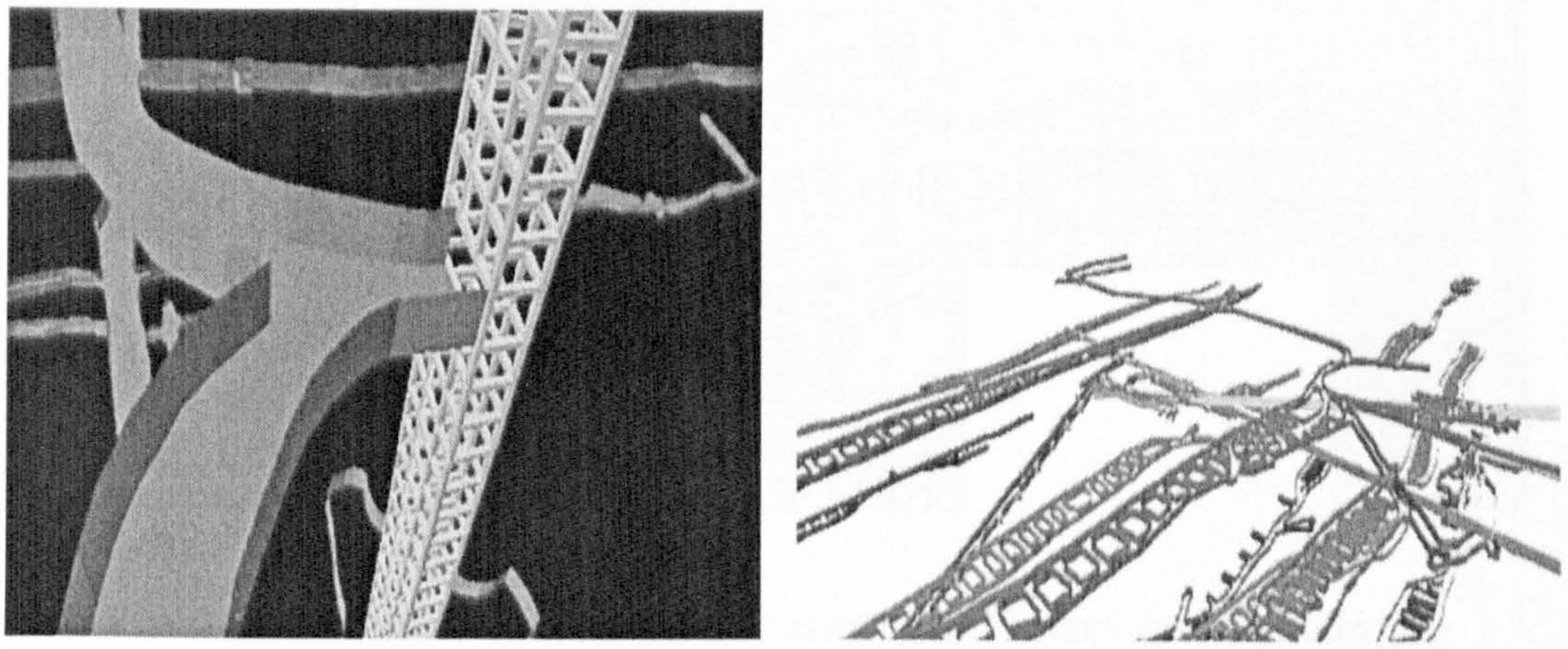


Figure 2-12 The view of a typical VRML tunnel outline along with the general view of the 400fm, 420fm, and 445fm levels

Martin Reddy et al. worked on designing and generating massive terrain data sets using VRML 97 (Reddy et al. 1999). To disseminate 3D maps and spatial data over the Web, They designed large terrain data sets accessible through either a VRML browser or the customised TerraVision II browser. Their design aimed at

efficient navigation of geo-referenced terrain data sets on the Web. Their research focused on the issues such as navigating models with many millions of polygons, managing deep level-of-detail (LOD) hierarchies, dealing with data precision problems to model the earth at sub-meter resolution, translating between geographic coordinate systems, and dynamically selecting different sets of geo-referenced data. Figure 2-13 shows a screen shot of the TerraVision system.

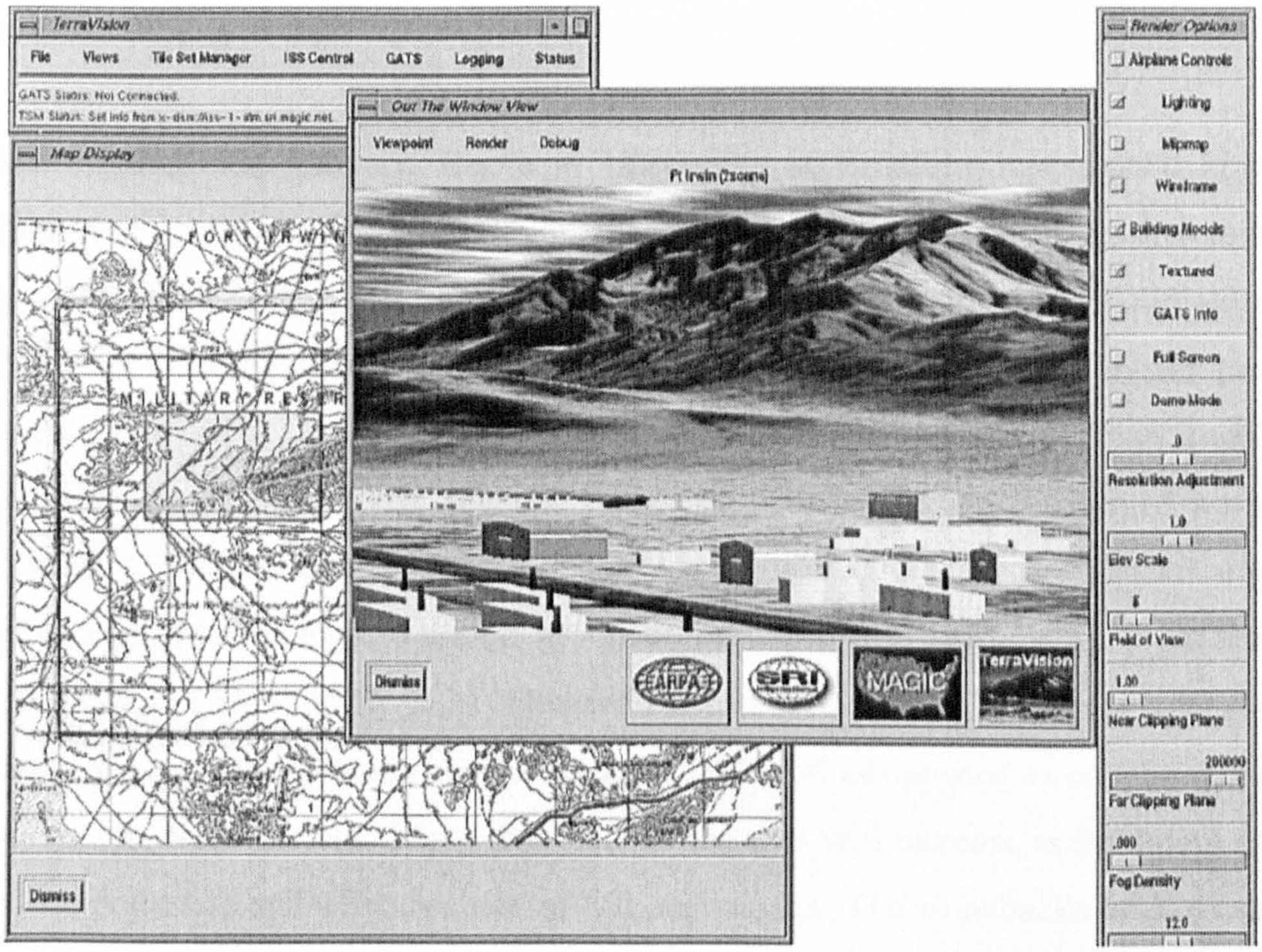


Figure 2-13 The screen shot of the TerraVision system

2.4.5 Collaborative working

As companies and government agencies continue to introduce distributed systems into the workplace, there is a demand for more sophisticated software tools to support different virtual organisations created. The aim of a virtual organisation is to provide support for members of one or more actual organisations who are

physically distributed or otherwise unable to meet (Hardwick et. al. 1996). **David England** et al. at University of Liverpool presented a virtual reality interface to a distributed cooperative system for government workers (England et al. 1998). Their work improved the levels of mutual awareness among colleagues by reinforcing the concepts of shared documents and showing some measures of sharing. However, there were some drawbacks with this approach. For example, the direct conversion from the original 2D arrangement data to 3D layouts can produce messy displays. In addition, the users may find it hard to relate their original objects to their 3D counterparts.

Ian Palmer and **Carlton Reeve** at University of Bradford investigated how theatre set designs could be communicated over the Internet to designer, directors and producers (Palmer and Reeve 1998). Their research examined collaborative design of theatre sets over the WWW. The use of VRML to model the Theatre has proved extremely valuable. It has been used to assist the design of sets in a real production and has greatly reduced problems with set design performed by a number of the production team in remote locations. A project with a similar aim of using VR technology to aid the production process of a performance event has since been carried out by **IBM Theatre Project** (IBM 2001). This was a large-scale project that allowed virtual immersive rehearsal as opposed to collaborative set design. Such projects imply that work in the area will increase in the future as it is a valuable and effective use of VR techniques. The drawbacks with these systems were the relatively simple interaction supported by VRML Script. It would be improved using Java in order to achieve much more natural interaction and to provide a greater range of information.

2.4.6 Medical training through WWW

Surgical training is a long and expensive process. **Ken Brodlie** et al. in Leeds University worked on the use of web-based virtual reality to offer a cheaper, yet still effective approach to surgical training (Brodlie et al. 2000). Two applications

of VRML surgical simulation were prototyped — one in neurosurgery, the other in vascular surgery. Similarly, the Manchester Visualisation Centre (MVC) has been collaborating with the Department of Neurosurgery at the Leeds General Infirmary (LGI) to promote the use of VR technologies for surgical training. **Nigel W. John** et al. (John et al. 1999) addressed this task by utilising VRML and Java to build a series of cost-effective surgical simulators that support 3D interaction and have wide availability on the Web. Problems with these works centred on the improvement of realistic model representation, precise simulation, and efficient interactive modes.

2.4.7 Summary

This section has outlined a literature review of the latest developments of VR on the Internet. The previous research and related projects were reviewed including: commercial applications, online education and entertainment, information visualization, data mining, collaborative working, and medical training. Such projects imply that work in the area will increase in the future as it is a valuable and effective use of VR techniques on the Internet. The drawbacks with these systems were the relatively simple interaction and low performance. However, it has been shown that traditional VR technology is gradually being applied on a broad range of Internet-based applications with the advancement in both of software implementation and network hardware. The combination of VR and Internet has become a great area of many potential research and applications for collaborative e-service through the WWW.

2.5 A literature review of VR in design and manufacturing

Virtual Manufacturing (VM) takes advantage of Virtual Realty (VR) technology in order to provide an e-manufacturing solution in a 3D visual and interactive way. It has shown us the powerful potential of VR by being combined with the latest generations of CAD/CAM tools to aid the product development process. VM has made VR no longer state-of-the art but rather an innovative technology to support modern industry (Grebner and May 1995) (Astheimer et. al 1995) (Bauer et. al. 1995). Engineers and mechanists have applied VM in a wide range of design and manufacturing applications including: product design and prototyping, facility layout design and visualisation, assembly process planning and simulations, operations of machines training, etc. This section examines the developments of VR in design and manufacturing.

2.5.1 Product design and prototyping

The CAD product design process is a continuum, starting with the initial "concept" stage, followed by detail design and ending with the prototype. VR has been beneficial in the prototyping stage of the CAD process (Stucki et. al. 1995) (Dai and Gobel 1994) (Dai et. al. 1996). Many companies such as **Northrop** or **Rockwell International** work on replacing physical prototypes with virtual ones for shortening the cycle of product development (Kraftcheck et al. 1997).

Researchers at **Caterpillar Inc.** have used VR to improve the design process for heavy equipment. **Dave Stevenson** and **John Bettner** in collaboration with the staff of National Centre for Supercomputing Applications (NCSA) developed a system that allowed them to quickly prototype wheel loader and backhoe loader designs (Ressler 1994). In particular, this system used a Silicon Graphics computer to generate the real time graphics display and to simulate the operation of the equipment. The project was awarded the 1993 NCSA Industrial Challenge

Award for VR Use. Figure 2-14 illustrates an operator driving the virtual equipment at the NCSA VR lab. This project had the advantage of allowing engineers to operate the equipment and evaluate visual obstructions in a natural manner without having to build a physical prototype. It took six to nine months to build full-scale models using conventional design methods. However, using the virtual reality approach, designs can be done in less than one month. Therefore, VR technology dramatically reduced costs and shortened the amount of time that is required to analyse a new design concept and incorporate it into the production process. However, it would need further research to provide more realistic representation and an efficient interactive way.

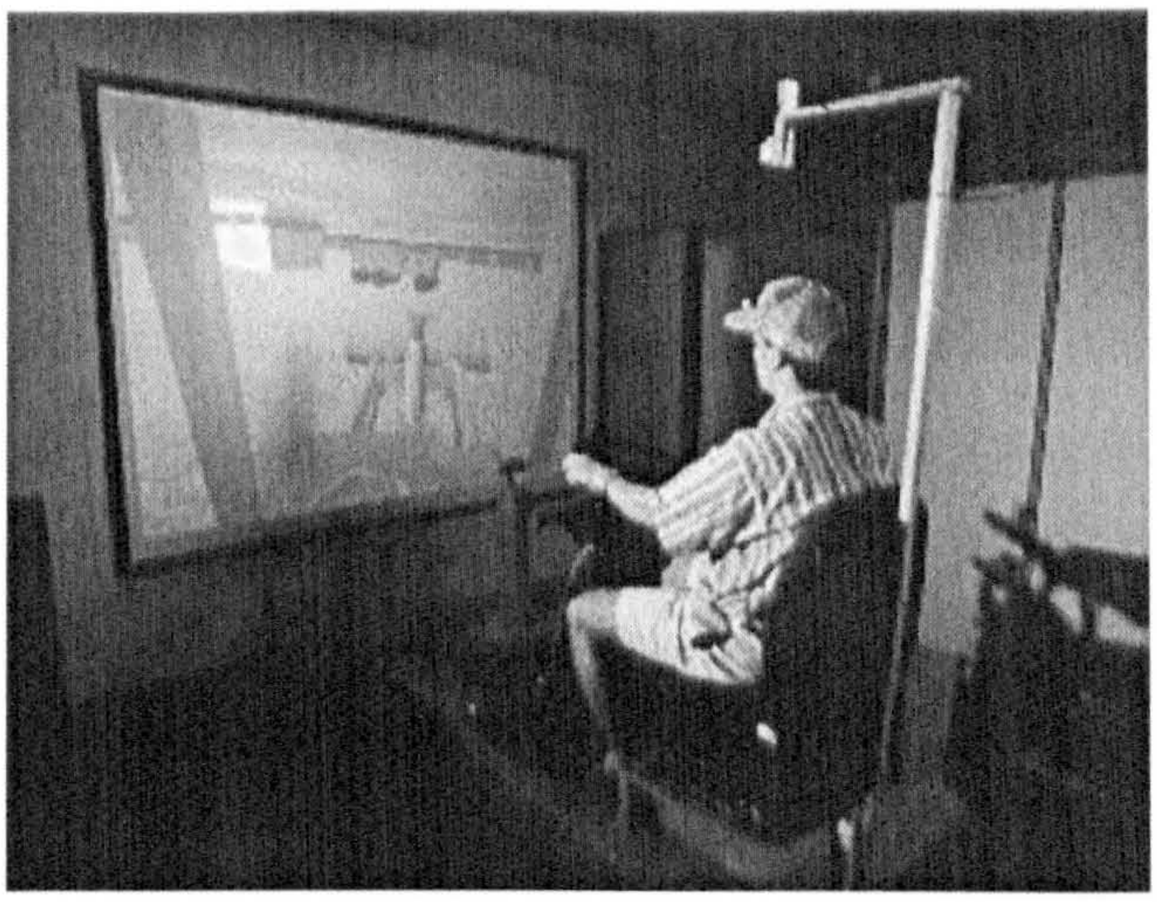


Figure 2-14 An operator driving the virtual equipment at the NCSA VR lab

In a manufacturing enterprise, virtual environment can improve productivity by allowing employees to train without taking production equipment out of service.

Figure 2-15 A flight controller uses the HST VR training system.

J. Wilson et al. at University of Nottingham (Wilson et al. 1995) reviewed the use of rapid prototyping as well as training by using lower-cost PC-based VR systems. **Ronald R. Mourant** et al. at Northeastern University, Boston, developed a virtual environment for simulating overhead crane operation (Mourant et al 1995). The crane simulator allowed an operator to manoeuvre a virtual cane for

improving training and reducing costs. In addition, the astronauts in preparation for Hubble Space Telescope (HST) repair and maintenance missions received training by using a virtual model of the HST (Loftin and Kenney 1995). Figure 2-15 shows a flight controller using the HST VR training system. These researches proved that virtual reality as a valuable training aid would play an important role in industrial applications. However, they did not facilitate to serve multiple users at the same time.

2.5.2 Facility layout design and visualisation

Manufacturing facility layout design is the process of determining the location of process and material handling equipment, work cells and the aisles for access within the limitations of the production plant facilities. In order to complete a facility design, it is necessary to determine how well the given layout will fulfil the needs of the production facility. Needs and constraints include material and personnel flows, electrical power and other resource availability, interactions between manufacturing processes, aesthetics, security and safety. To address all of these concerns, the representation and direct manipulative capabilities of VR technology seem particularly valuable for visualising and assessing complicated production facility layout.

M. Kreitler, J. Heim and **R. Smith** at University of Washington applied VR in the design and analysis of production facilities layout (Kreitler et al. 1995) (Smith and Heim 1999). The number of possible layout alternatives and the need to accommodate more than 2D manipulation of work pieces and tooling indicates that a mathematical approach is unlikely to be sufficient for determining machinery and equipment positions as well as evaluating the efficiency of work-in-process, materials and tooling movements. Their research made rearrangement of the VR shop floor as easy and natural as possible for user. It emphasised the advantage of 3D manipulation in facility layout design and visualization but there was still a lack of rich real-time interactive capabilities in their system.

Robert P. Smith and **Joseph A. Heim** worked on applying VR technologies in manufacturing facility design (Smith and Heim 1999). The purpose of their study was to determine the appropriateness and utility of applying VR technologies in the layout design of manufacturing facilities. They constructed an interactive, 3D environment to aid the facility designer. The system development demonstrated that an interactive, 3D display could add significant value to the facility layout design process. Their research has shown that for manufacturing environments where the third dimension is critical to system performance, interactive 3D display systems are better able to present the information needed by facility designers.

However, these works did not provide a network environment, which would benefit the multiple users that are distributed in different places. The lack of widespread communication networks is a weak point within a collaborative simulation environment.

2.5.3 Assembly process planning and simulations

Assembly planning determines how parts are put together into an assembly. An assembly sequence is the foundation of many production engineering activities, including line balancing, facility layout, and production system design. Thus, an assembly sequence has a major impact on production efficiency and cost. Many products contain hundreds of parts, and such complex assemblies have presented considerable difficulty in assembly planning to production engineers.

To support assembly planning, **N. Ye** and **F. Dech** at University of Illinois carried out a study on the assembly planning effectiveness by using VR (Ye and Dech 1999). They used three environments — a traditional environment using blueprints, a non-immersive desktop VR environment, and an immersive projection-based VR environment to teach participants skills for assembly planning through examples, and hence measure the effectiveness of their learnt

skills in solving a different example problem. The results of experiments have shown that the participants can perform the assembly operation in approximately half the time in the immersive and non-immersive VR environments than in the traditional environment. This project has shown that VR as an innovative technology can improve training and work efficiency and further increase productivity. On the other hand, it justified the need for further research to be carried out in this field.

The assembly of aircraft is very complex task that is difficult to automate. Many of the skills required demand dexterity that not easily accomplished by robots. In addition, aeroplanes consist of many small-lot size parts and reprogramming robots for these quantities is an expensive prospect. Thus, **Boeing** uses VR to aid the assembly of aircraft (Gaudell and Mizell 1992).

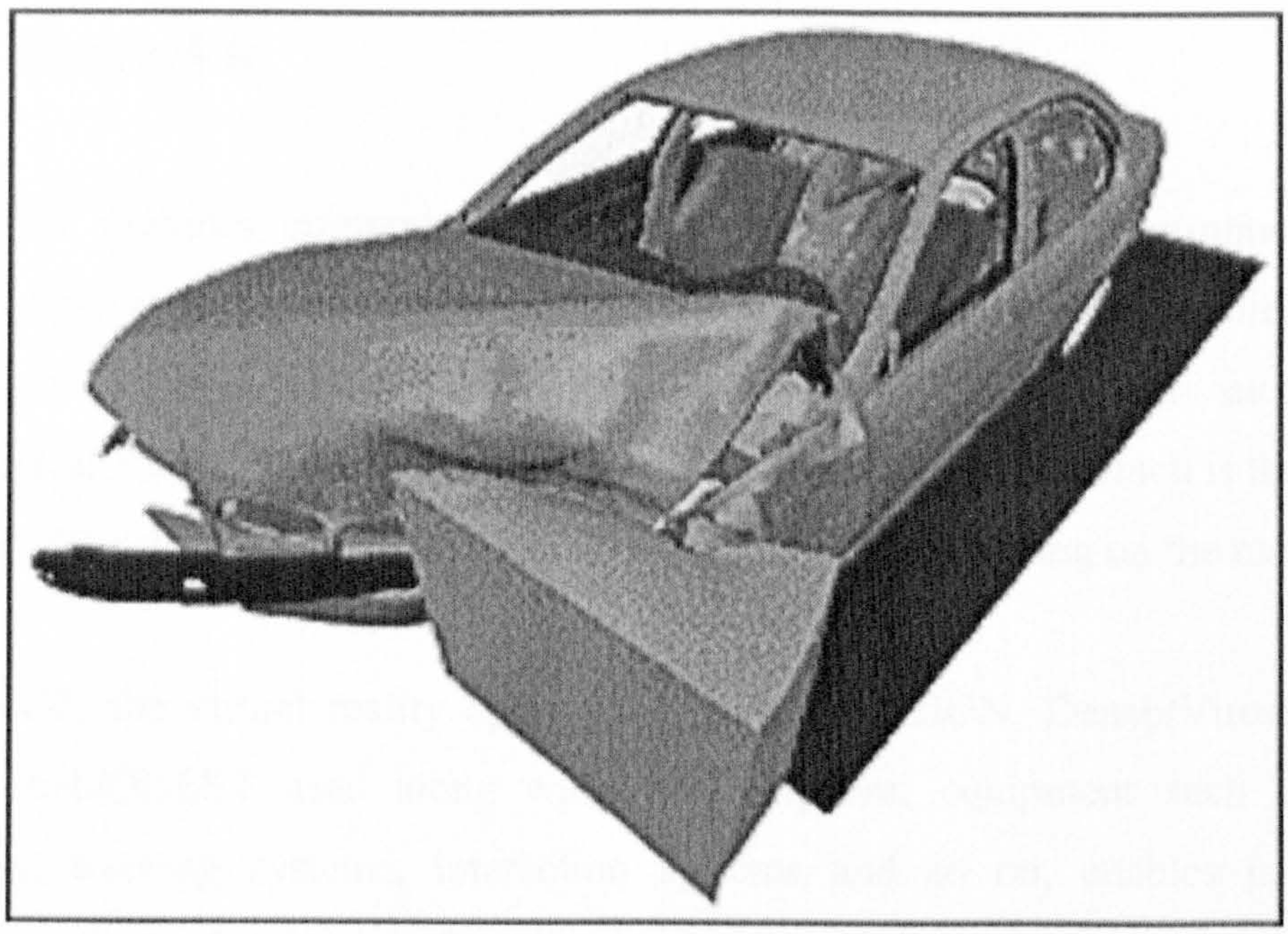


Figure 2-16 The car crash-test simulation in the virtual environment

Martin Schulz and **Thomas Ertl** at University of Erlangen in co-operation with **Thomas Reuding** at BMW carried out a study on the analysis of engineering simulations in a virtual environment (Schulz et al. 1998). They developed a virtual

environment for car-body engineering applications. Their system is being used at BMW to analyse a wide variety of time-dependent numerical simulations, ranging from crash-worthiness to sheet-metal forming, vibrations and acoustics. Figure 2-16 shows a crash-test simulation in the virtual environment. Users can visualise geometry data and physical property data like stress, strain, acceleration, or velocity to illustrate the performance of the whole vehicle. Its main advantage is the intuitive navigation and interaction with different types of Finite-Element Models (FEMs) relevant to the car-body development process. However, this research still needs to continue working on the integration of different simulation types into the system, more intuitive interaction metaphors, and scalability of the system to run on midrange computing hardware.

2.5.4 Commercial virtual manufacturing (VM) systems — Deneb/VR

Since VR provides comprehensive analysis of systems and environments to reduce risk, cost, and lead times, some well-known industrial companies have begun to develop commercial VR software products. One of successful commercial Virtual Manufacturing (VM) systems is Deneb/VR, which is the most popular VR solution for applications in design and manufacturing on the market.

Deneb/VR, the virtual reality option to Deneb/ENVISION, Deneb/Virtual NC, and Deneb/QUEST used along with VR peripheral equipment such as VR displays, tracking systems, interaction systems and so on, enables users to immerse themselves into Deneb simulations for enhanced visualization, evaluation, and virtual operations.

Deneb's Deneb/ENVISION provides a superior physics-based, 3D environment specifically for designing, verifying and rapid prototyping of concept designs involving structures, mechanical systems and humans. It enhances systems and subsystem level models with physics-based motion, virtual reality immersion, and

ergonomic evaluation capabilities for highly accurate 3D simulation, analysis, visualization. Deneb/ENVISION VR takes advantage of VR technology to provide a powerful environment for designing and programming complex design, engineering, and training operations by allowing the user to interact more directly with the simulation environment. Users can "fly" and "walk through" the simulated environment using VR and telepresence techniques to operate devices, alter the environment by relocating components, or otherwise interact with the virtual world. Deneb/ENVISION VR can support the following features: alternative display technology; geometric representation of objects; interaction between immersed objects and simulated world; position tracking and event handling; 3D spatial sound. In addition, a wide range of visual display systems, position and object tracking systems, and event feedback devices are supported by this system.

Deneb's Deneb/Virtual NC is an interactive 3D simulation environment specifically designed for visualizing and analyzing the functionality of a machine tool, its CNC controller, and the material removal process. It is being used to prove out NC programs, setup configurations, advanced machine tools and complex machining processes. Virtual NC provides a dynamic environment for concurrent engineering, rapid prototyping, machine tool and machining simulation, and operator training. With the introduction of VR, Deneb/Virtual NC has been widely used to train new machine operators and NC programmers. For advanced training, operators and programmers can gain proficiency with high-level CNC controller commands safely, quickly, and without risk by avoiding equipment crash. Thus it provides a safe environment through which engineers and programmers can evaluate the actual machine controller and different processing techniques without monopolizing machine time or risking damage to the actual expensive machine tool.

Deneb's Queuing Event Simulation Tool, Deneb/QUEST represents a quantum leap in the modelling and analysis of manufacturing systems. Detailed physical system properties combined with VR technology and visual analysis deliver a new

level of ease, power, and accuracy. It gives new meaning to "interactive modelling" and "what if" evaluations. Dynamic behaviour animation for event simulation can be generated in real time. In addition, Deneb/QUEST includes a rich resource library of geometric models which contains conveyors, workcells, robots, cranes, etc. Other devices can be built with other CAD package or imported from IGES, DXF format. Model states can be saved and recalled at any point during the simulation. Deneb/QUEST incorporates real production variables, such as physical lengths, speeds, accelerations/decelerations, and plant layout to analyze the effects on material handling equipment and labour. For example, Deneb/QUEST simulations enable parts of different sizes to accurately accumulate on conveyors, and user-defined labour paths to determine the time spent by a human model walking to service a workcell process.

Deneb/VR as a commercial VR product has been successfully used for concurrent engineering, customer evaluation of new products, mission rehearsal, and training. However, the current Deneb/VR solution still lacks multiple-user collaborative working support via a network environment although the Deneb Company has begun to develop Virtual Collaborative Engineering (VCE) in order to complement and meet the demand of collaborative e-service in design and manufacturing.

2.5.5 Summary

This section carried out a literature review of previous and related VR research and projects in design and manufacturing including: product design and prototyping, facility layout design and visualisation, assembly process planning and simulations, operations of machines training, etc. It has been demonstrated that VR has been applied in design and manufacturing for increasing productivity, speeding decision making, reducing costs, and improving time to market by eliminating costly and timely travel aspects of the review processes. Also improved communications between engineers, clients, suppliers, and/or product teams prior to product launch reduce the need for multiple revisions and

prototypes. However, it has also shown that most of current VR applications in design and manufacturing are often expensive due to the involvement of high performance graphics accelerators, multiprocessor graphics workstations and some dextrous immersive devices. Each workstation usually needs dedicated licensed software and maintenance supports. The evidence has shown that there is a gap between the Web-based VR and industrial applications. Little research work had been done to apply Web-based VR technologies to support multi-users collaborative working in design and manufacturing via the Internet.

2.6 Internet-based collaborative product commerce — Windchill® system

Optimizing product development requires a collaborative effort among employees, customers, and all members of the supply chain. This collaboration means that information has to be readily accessible to all those involved so decisions can be made jointly and quickly. Industry analysts such as GartnerGroup, Aberdeen Group, and CIMdata have been observing and analyzing recent changes in business dynamics, and they agree that a new Information Technology (IT) market is emerging, which is called Collaborative Product Commerce (CPC). CPC enterprise solutions are Web-based solutions that use the Internet to allow employees, customers, and suppliers to collaboratively develop, build, and manage products throughout the lifecycle. These industry analysts have concluded that manufacturers who make CPC one of their primary enterprise business initiatives will emerge as leaders in their markets.

2.6.1 Enabling collaborative product commerce

Product information is not only dispersed throughout the organization, but all over the globe through a worldwide network of suppliers and business partners. All of these organizations potentially have their own process and applications for managing product information. Managing product and process lifecycles has

taken on global proportions. The challenge has now become how to link up quickly with a vast number of organizations, departments, and business partners all over the world in order to create an environment for collaborative product development. CPC solutions leverage product knowledge to drive innovation and expand business opportunities. CPC solutions will realize following:

- **Enable inter-enterprise collaboration**

  Meeting the demand for custom-tailored products requires that customers, suppliers, and employees collaborate together to jointly develop and manage products throughout the lifecycle. Customers enter the development process early to help design products, and product information is shared with customers and suppliers to reduce the number of design iterations and bring the product to market faster. CPC solutions help companies enable the collaboration process by creating an environment where product information can be shared and exchanged across organizational boundaries.

- **Make companies more agile**

  Companies are continually adjusting their mix of suppliers and partners opportunistically to take advantage of new technology, product innovations, domain expertise, and better business relationships. CPC solutions make companies more agile by allowing them to connect instantly to the heterogeneous processes and business systems of each new supplier so that information can be accessed and exchanged readily.

- **Manage the entire product lifecycle**

  Manufacturers must consider how products are created, managed, and evolved during all phases of the product lifecycle, from concept and definition to sourcing, production, service, maintenance, and retirement. Until now, there has not been a complete solution for managing the entire product lifecycle. Point solutions focus on part of the problem, but without a strategy to knit these stand-alone systems together, information remains isolated and inaccessible. The most successful CPC solutions will provide

a broad range of functionality to address the entire product lifecycle. They will also allow manufacturers to seamlessly link distributed data sources to create a common representation of the product as it evolves over time.

2.6.2 Windchill solution for Collaborative Product Commerce

PTC has made CPC one of its primary enterprise business initiatives from now on. PTC's Windchill product has begun to be designed to focus on the lifecycle of products from concept and definition to production, service, maintenance and retirement. It advances CPC initiatives by allowing manufacturers to collaborate over the Internet with their customers, suppliers, and partners throughout the product development and delivery process. It does this through its unique web-based approach and federated architecture (PTC 2003):

- **Windchill's Web-based approach** — can enable customers, partners, and suppliers to collaborate in a Web environment to create innovative new products, deliver those products to market faster, and capitalize on enterprise expertise, customer knowledge, and supplier innovation while accelerating the global introduction of innovative products to market with reduced risk and lower costs.

- **Windchill's federated platform** — provides accurate, up-to-date information to all enterprise participants via a Web browser, so they can jointly make decisions and promote products through each stage of the lifecycle.

Based on the Windchill federated, web-based architecture, this application set provides an e-business solution to product and process lifecycle management for CPC. This solution creates an inter-enterprise collaborative environment for the sharing and visualization of product and process knowledge, regardless of where the information resides or what format it is in. The Windchill collaboration system

transforms product knowledge into a significant enterprise business asset by making it available to the extended enterprise.

2.6.3 Summary

PTC's Windchill system provides a comprehensive suite of e-business solutions for the emerging collaborative product commerce (CPC) market. It enhances the value of existing IT investments by connecting and leveraging information among business systems. It facilitates customer, supplier, and partner collaboration to deliver innovative products through the Internet. As companies must be able to respond quickly to their dynamic business environment, the Windchill system supports a collaboration environment by providing companies with access to the product and process information they need to make the best business decisions, and this allows them to identify new market opportunities and drive new levels of competitiveness. The benefits and limitations of the Windchill collaboration system will be further discussed in details in Section 7.3, Chapter 7.

2.7 Conclusions

At present, competitive pressures are getting stronger in a global marketplace that demands better products, in a short time and at a lower cost. Therefore industrial organisations need to develop optimum product designs, manufacture and test these products using the state of art VR technologies, and then market them faster, less expensively, and more effectively than before. With the rapid growth of network technology and Internetworked 3D graphics techniques such VRML/X3D, it is possible to extend low-cost VR applications on the Web. Thus, Web-based VR is an enormously advantageous tool for expediting insight into complex problems, reducing production costs, increasing production output, and enabling collaborative decision-making, all leading to the avoidance of costly mistakes prior to production commences. In addition, although CPC enterprise

solutions are Web-based solutions that use the Internet to allow employees, customers, and suppliers to collaboratively develop, build, and manage products throughout the lifecycle, the current CPC solution like Windchill solution mainly transfers and distributes data and information in the form of text, document, product structure and images. Such traditional methods of information representation present severe limitations due to the lack of innovative information representation and visualization elements on the Internet which can be dramatically enhanced by integrating with Web-based VR technology.

In order to allow industries to utilise VR and CPC together sufficiently and cost-effectively, there is a demand to investigate Networked Virtual Environments (Net-VE) and to propose an approach to apply Web-based VR technologies for supporting interactive applications in design and manufacturing, which this research project is targeting. According to the approach, a distributed VR-based system is required to be designed and implemented by combining VR and network technology in order to serve multiple users such as the SMEs distributed all over the world for collaborative working.

Chapter 3

Design of a Cost-effective Web-based VR Approach in Net-VEs*

3.1 Introduction

Networked Virtual Environments (Net-VEs) have already begun to foster an insightful, intuitive and interactive system that allows effective communication among multiple users for e-service. With the dramatic change of network technology over the past few years and the increasing speed of the Internet, The Internet has become a feasible network for Net-VEs deployment. This chapter presents the design of a cost-effective Web-based VR approach to support interactive applications in design and manufacturing via the Internet for collaborative e-service.

Section 3.2 firstly presents the concept and features of Net-VEs. Section 3.3 carries out a survey of Net-VE systems by reviewing previous research and projects and then Section 3.4 analyses the major challenges in Net-VEs.

* The main content of this chapter has been published in the proceedings of **IEEE SMC 2001**, IEEE Systems, Man, and Cybernetics International Conference, Oct. 7-10, 2001 at Arizona, U.S.A (This paper was awarded **the Best Student Paper Finalist.**)

Section 3.5 proposes a cost-effective approach to apply Web-based VR for supporting design and manufacturing via the Internet. The distributed VR system in a Net-VE by using the WWW is designed by the author for interactive applications in design and manufacturing in order to benefit multiple users, especially for SMEs for collaborative e-service. The detailed architecture of the Web-based VR system is then presented. Finally, this section discusses three potential applications that can be prototyped based on the proposed approach. They respectively concern three applications: machining, process planning, and factory layout.

3.2 Networked Virtual Environments for E-service

E-services are complex distributed applications that can be accessed by other applications or components over the Internet and across organizational boundaries. E-services are self-contained and modular-based. A complex distributed e-service can be modelled as several service tasks. Each task has its own resource and timing requirements and generates a result that triggers the execution of the subsequent task. A service task is defined as a sequence of method invocations of objects distributed across multiple processors in multiple domains. E-services can be composed and deployed dynamically. As an example, E-manufacturing as a new generation of product development solution allows manufacturers all over the world to speed up and slim down everything from design to manufacturing through the Internet. It has been employed in a wide range of manufacturing activities.

Networked Virtual Environments (Net-VEs) have already begun to foster an insightful, intuitive and interactive system that allows effective communication among multiple users for cooperative work (Benford 1995). This section presents the concept and features of Net-VEs.

3.2.1 The concept of Net-VEs

Virtual Environments (VE), also referred to as Virtual Reality (VR), is an interactive 3D computer graphics technology allows people to visualize and interact with the computer generated environments in real time. Thus, Net-VEs should provide users with an adequate level of realism; create a sense of presence along with a fully or partially immersive feeling and real time interactive experience from high performance computer graphics and stereo sound in a networking environment. Recent research has shown the Net-VEs' powerful potential. For example, **C. Flerackers** et al. created an interactive drama in a Net-VE (Flerackers et al. 2001). This project developed ten episodes of an interactive television drama series, which allowed children to explore opportunities to participate using a networked virtual environment. It bridged the gap between the conventionally passive medium of television and the interactive medium of Internet. Moreover, most networked games such as Ultima Online (Ultima Online 2001) have served as inspirations to the Net-VEs community. They have become more and more popular in the game market by taking advantage of Net-VEs to attract multiple users.

Due to the involvement of the multiple independent users, it is obvious that Net-VEs not only serve for the single user as the standard virtual environments, but also benefit more users and applications. In addition, the ability to share 3D virtual worlds differentiates Net-VEs from traditional Virtual Networking Computing (VNC) in the form of networked meeting, chat rooms, and E-mails; and the ability of real-time interaction differentiates Net-VEs from typical Web Browsers. Net-VEs should be an entity by integrating the following three areas for E-service.

- **Real-time graphical applications:** To ensure the implementation of graphics rendering pipeline and maintain real-time display frame rates (at 30 frames/second). The graphics render pipeline consists of three stages — application, geometry and the rasterizer. The application stage may contain collision detection, speed-up techniques, animations, force

feedback etc; the geometry stage deals with transforms, projections, lighting, etc. the rasterizer stage draws (renders) an image with use of the data that the previous stage generated (Moller and Haines 1999).

- **Interactive applications:** To process real-time data input from users. Users should see the virtual environment as if it exists locally, even if its participants are remotely distributed and across the network.

- **Distributed systems:** To manage network resources and cope with data loss and synchronization. The computations at the clients involved must be synchronized.

3.2.2 The features of Net-VEs

According to **S. Singhal** and **M. Zyda,** a Net-VE system should have the following common features (Singhal and Zyda 1999):

- All participants have a sense of being in the same space, for example, in the same workshop, laboratory, or building. The shared world must present the same characteristics to all participants such as surface properties, dynamic properties, physical constraints, acoustic properties and even illumination model.

- All participants should get a sense of presence when entering a Net-VE. They have the illusion that they are in the virtual world. All of them can manipulate virtual objects. They have the same abilities and behaviours such as picking-up, moving, dragging and so forth.

- Not only should participants be able to navigate the shared virtual world at the same time, but also they can interact with the shared virtual world in real time.

In addition, it is necessary for a Net-VE to provide a common communication facility. This allows participants to communicate with each other and exchange information in time. It can be implemented in the form of typed text or voice.

3.3 A survey of Net-VE systems

The growth of Net-VEs has seen a rapid rise in the development of systems that allow a number of users to share a 3D virtual space. These range from research prototypes to commercial products and include the following systems and applications. This section carries out a survey of Net-VE systems by reviewing previous research and projects.

3.3.1 SIMNET and DIS

SIMNET (Simulator Networking) is a distributed military virtual environment originally developed for the Department of Defence, US, by Bolt, Beranek and Newman (BBN), Perceptronics, and Delta Graphics. The SIMNET aimed at a low-cost Net-VE for training small units such as M1 tanks, helicopters, command posts, and so on to fight as a team.

In order to build high-quality, low-cost simulators and create a consistent networked virtual battlefield, the SIMNET project set up an 11-site test-bed with from 50 to 100 simulators at each site. SIMNET could be accessed from anywhere on the network using a simulator as the portal into the synthetic environment. The online multiple users could interact with each other in the sharing synthetic battlefield.

The software architecture of a typical SIMNET node is made up of the main modules, as shown in Figure 3-1. There are three basic functional characteristics involved in the network software architecture (Singhal and Zyda 1999):

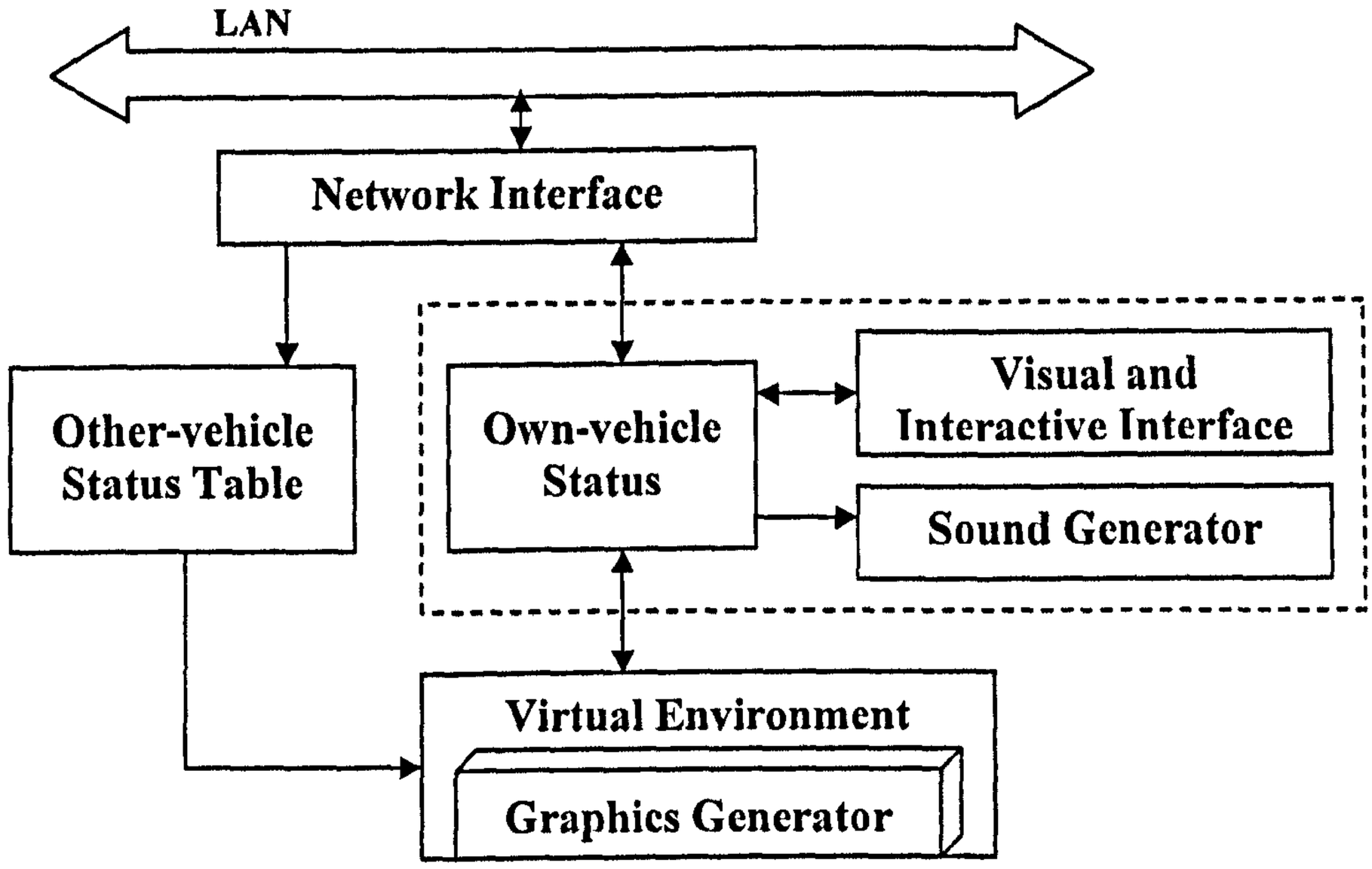


Figure 3-1 SIMNET software architecture

- **The object-event architecture:** The architecture modelled the world as a collection of objects whose interactions with each other are a series of events associated with the changes in world or object state. Objects (e.g. vehicles, weapons) could interact across the network. The architecture handles with events in SIMNET by message passing. For example, an event would be transmitted via a message to indicate that the change of the position of a tank.

- **Autonomous simulation nodes:** The mechanism allows each user, vehicles, and weapons systems in the synthetic battlefield accurately represent their current state onto the network by passing messages. Each node is responsible for one or more objects in the virtual world. This means that the node is able to pass messages onto the network representing the current state or any change-in-state of its objects. If objects that change frequently or rapidly in state, the corresponding node has to pass large numbers of messages onto the network.

- **Predictive modelling algorithms:** A well-defined set of predictive modelling algorithms called "dead reckoning" algorithms were embedded in system for reducing the message traffic caused by the large number of messages that flooded the network and overloaded the CPUs from multiple users.

Due to SIMNET's enormous success in military simulation training of U.S Army, the network software architecture of SIMNET protocol was further evolved into the Distributed Interactive Simulation (DIS) for general purpose. The DIS (Locke 1997) intends to become IEEE 1278 standard that allows any type of user to participate in the distributed interactive simulation virtual environment on any type of machine. Based on the non-proprietary and well-documented underlying network software architecture, DIS was a successful solution for the fully distributed, heterogeneous, and large-scale simulations for various applications among a greater variety of users. The success of DIS led Net-VEs from being only of military concern to having more widespread availability and interest.

While workstation-based VEs are some of the earliest inspiration for Net-VEs, the PC has taken the desire and interest for such connected worlds to the next level. The networked games have inspired research in the Net-VE community. The networked games have made Net-VEs widespread and well-known among ordinary people, meanwhile they have also facilitated the evolvement of the new technologies involved in Net-VEs. Many influential games made people realize and accept Net-VEs. For example, the popular game — *Doom,* released by id Software in 1993, was able to provide online gaming networks. It allowed players to access a believable 3D environment to fight against the monsters represented by the other online players. Without dead reckoning and flooded LANs, the networked ability of *Doom* has allowed multiple users to achieve a sense of presence and real-time interactivity in a 3D synthetic environment. Furthermore, its success created enormous demand for more 3D networked games.

3.3.2 NPSNET

NPSNET (Macedonia et. al. 1995) is a distributed networked 3D virtual environment system developed by the Computer Science Department of the U.S. Naval Postgraduate School (NPS). It is designed to support large-scale military training and simulation. The NPSNET Research Group (NRG) conducts the longest continuing academic research effort in Net-VEs. The group focuses on the complete breadth of human-computer interaction and software technology for implementing Large-scale Virtual Environments (LSVEs).

The NRG has produced several generations of NPSNET. Since the NPSNET-1 was developed in 1990, NPSNET-2 and -3 were enhanced with faster ways to produce graphics and extend the size of the terrain databases. NPSNET-Stealth derived from NPSNET-1 in 1993 with the goal of developing a system capable of reading SIMNET terrain databases and SIMNET networking protocols. NPSNET-IV (Macedonia et al 1994), based on the Distributed Interactive Simulation (DIS) protocol, took advantage of SGI's Performer library to create 3D graphical representation of virtual scenarios and incorporated dead-reckoning algorithms to reduce network traffic, and included spatial sound. As a result, NPSNET-IV has made a major contribution to the development of scaleable network architectures and interest management techniques. In addition, NPSNET-IV was the first Net-VE system to use IP Multicasting groups (Kumar 1995) to address a large number of users for large-scale virtual environments. The system is already able to support several hundred parallel users.

In NPSNET, the virtual environment consists of the landscape and a number of entities — 3D objects (tanks, fighters, bullets, bridges, buildings, etc.). Every entity has an ID that is unique in the world. For example, the tank might have an ID of 20, the helicopter might have an ID of 41, and so on. Each entity has its own state variables, such as its current location (position, orientation etc.) The EntityMaster has to transmit these state variables to the EntityGhosts that represent the entity on other hosts. Each participant's machine can be configured

to simulate a military vehicle or dismounted infantry personnel, and large number of interacting 3D objects can be supported, as shown in Figure 3-2. While the landscape is set, each entity's state is distributed to all participants periodically by the multicast mechanism. Thus, new participants connecting to an ongoing simulation can catch up easily once they have loaded the virtual 3D scenario.

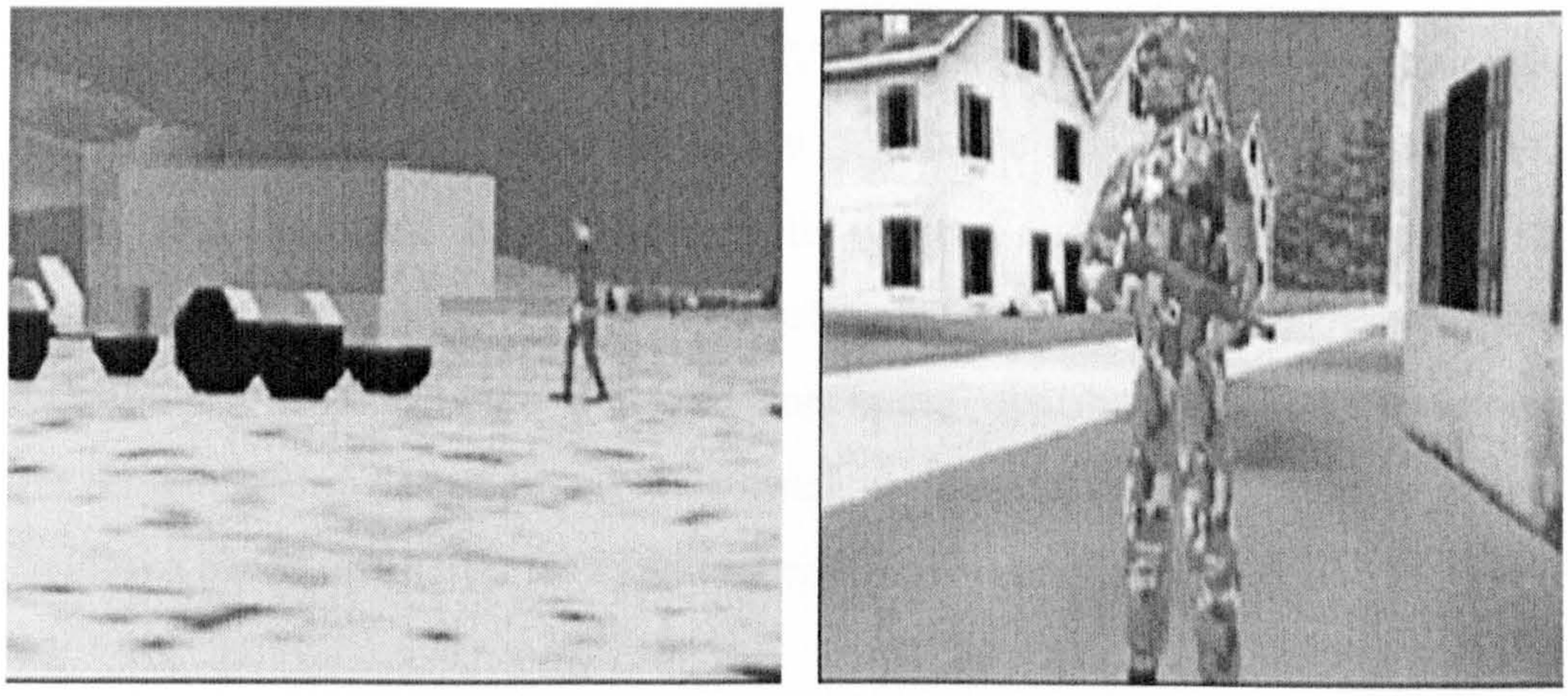


Figure 3-2 The virtual scenario for simulating battles and a virtual soldier in the NPSNET system which was developed by the U.S. NPS.

The evolution of NPSNET networking is shown in Table 3-1. In order to give the NPSNET system maximum reliability and interactivity for the specific task, NPSNET-V's networking intends to be flexible, dynamic, and comprehensive.

Versions	**NPSNET networking**
NPSNET-1, 2 & 3 (1992)	NPS invented protocol for LANs only
NPSNET-Stealth (1993)	SIMNET protocol and bridged LAN and WAN communications
NPSNET-IV (1995)	DIS protocol and IP Multicast for WAN communications
NPSNET-V (2000)	Virtual Reality Transfer Protocol (vrtp) and HLA for communications

Table 3-1 The evolution of NPSNET networking

3.3.3 DIVE

The Distributed Interactive Virtual Environment (DIVE) developed by the Swedish Institute of Computer Science is an Internet-based multi-user VR system where participants navigate in 3D space and see, meet and interact with other users and applications (Hagsand 1996). As one of the early academic Collaboration Virtual Environments (CVEs), DIVE has been continuously developed since the early 1990s and is now accessible from a typical desktop PC and can also be used immersively with a head-mounted display, or with projection-based displays. The DIVE software is a research prototype that provides multi-user software platform for many distributed VE applications. DIVE supports the development of virtual environments, user interfaces and applications based on shared 3D synthetic environments.

DIVE is especially tuned to multi-user applications, where several networked participants interact over a network. The aim of the DIVE is to allow distributed participants to be embodied in and freely navigate 3D graphical worlds and to communicate using a combination of real-time networked audio, video, text chat, simple gestures and meeting support tools such as shared whiteboards.

The DIVE project was originally developed as a *Multidraw* system in the early 1990, which is a shared 2D drawing editor. The system then became the beginning of the Telepresence Project. The first demo of Telepresence was *Virtual Clock* where a viewer on one machine could watch a simple 3D clock running on another machine. The program was implemented on IBM RS/6000 workstations with GL graphics accelerators. More advanced and complicated demos were developed in the mid of 1991. Telepresence was rebuilt during the 1992. The updated system used a new distribution mechanism and revised co-ordinate systems. Automatic behaviours were added into objects in the system. The project was renamed as DIVE2 later on in 1992. The new system was implemented on Sun XGL architecture so that it allows up to three participants to interact with the virtual world.

DIVE provides system architecture for implementing multi-user interactive virtual environments. The architecture focuses on software and networking solutions that enable high interaction at each participating site, i.e. interaction results are immediately shown at the interacting sites but slightly delayed at remote sites.

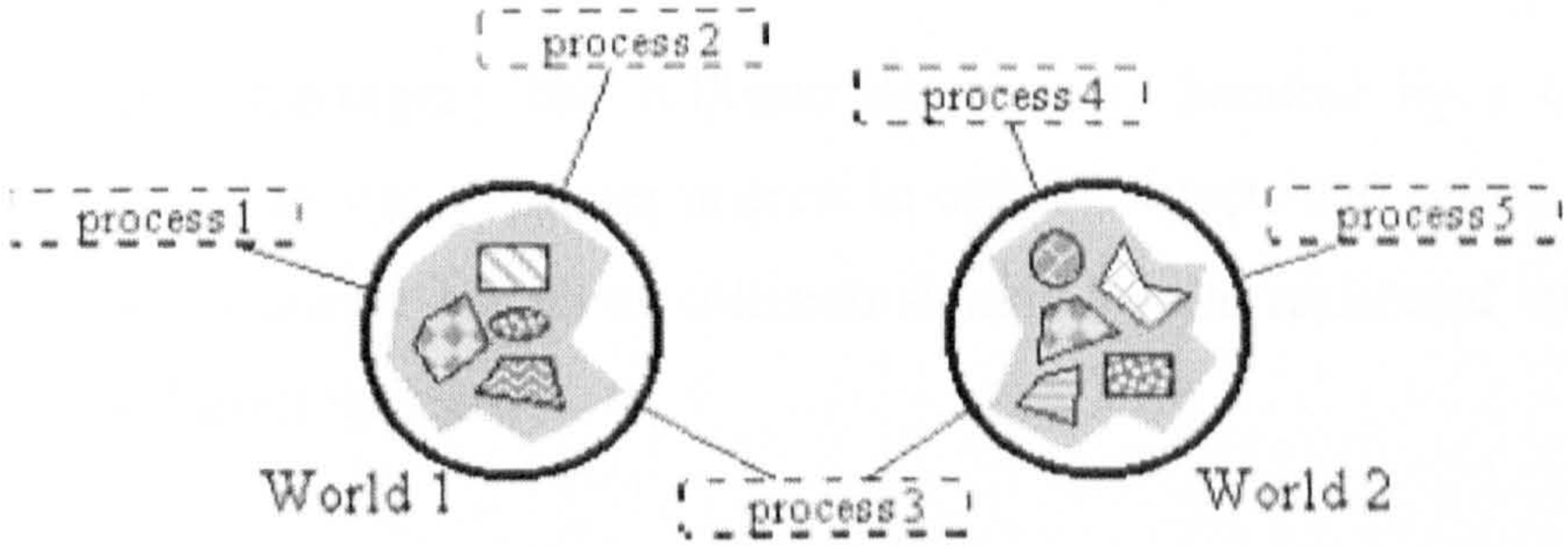


Figure 3-3 In DIVE, separate processes interface and interact through one or more distributed world databases. (Hagsand 1996)

A central feature in the programming architecture of DIVE is the shared, distributed world database. All user and application interactions take place through this common medium. Figure 3-3 shows five applications interacting through two different worlds. Application processes 1, 2, and 3 interact with each other through World 1. Applications 3, 4, and 5 interact with each other through World 2. Note that application 3 is interacting with both worlds.

The universe of the DIVE has following characteristics:

- **Scene graph hierarchy:** Each object carries the essential logical, interaction, and dynamic information. This includes geometrical orientation, material descriptions, and variables controlling interaction and rendering. Each object is composed hierarchically; its own geometrical transformation is composed with the rotation and translation of the object at the next level in the scene hierarchy.

- **Separate virtual space:** A world represents a separate virtual space disjoint from other worlds, with its own set of objects, actors, and view. It also defines a separate spatial domain and is therefore assigned a multicast address. Request of entities belonging to the world can be made by sending a message to the world's multicast address.

- **Collision manager:** The collision detection is handled by a *Collision Manager*. Processes register interest in certain objects and actors, and the collision manager generates collision signals for the registered entities as their volumes intersect.

- **Visualiser:** A process with a 3D rendering module that associates a real human user with a virtual user — a virtual actor that is called a *Visualiser*. *Visualiser* presents a graphical representation of a virtual world and lets the use interact with that world by selecting and grabbing objects, sending messages to actors, setting up audio connections, and so on. DIVE provides many *Visualisers*; each is suited for different interface requirements. For instance, with an immersive interface, magnetic trackers monitor body movements, and an HMD provides the worldview.

- **Behaviour simulation:** Dynamic behaviours of objects are described by Tcl scripts. Tcl is a portable and interpretative scripting language that can be executed immediately on any platform without compilation. Tcl scripts are triggered by events in the system, such as user interaction signals, timers, collisions, etc. DIVE reads and exports VRML and several other 3D formats. It is integrated with the World Wide Web.

DIVE uses a distributed, fully replicated database similar to that of SIMNET and DIS. The main difference is that the entire database in DIVE is dynamic and extensible, which means that new objects can be added and modified in a reliable and consistent way. The system achieves this by employing a multicast protocol and concurrency control via a distributed locking mechanism. However, this

method imposes significant data communication burden on the net. This results in the difficulty to scale the DIVE to more than 16 users.

DIVE applications include virtual battlefields, spatial models of interaction, virtual agents, real-world robot control and multi-modal interaction. Some snapshots of these applications are shown in Figure 3-4.

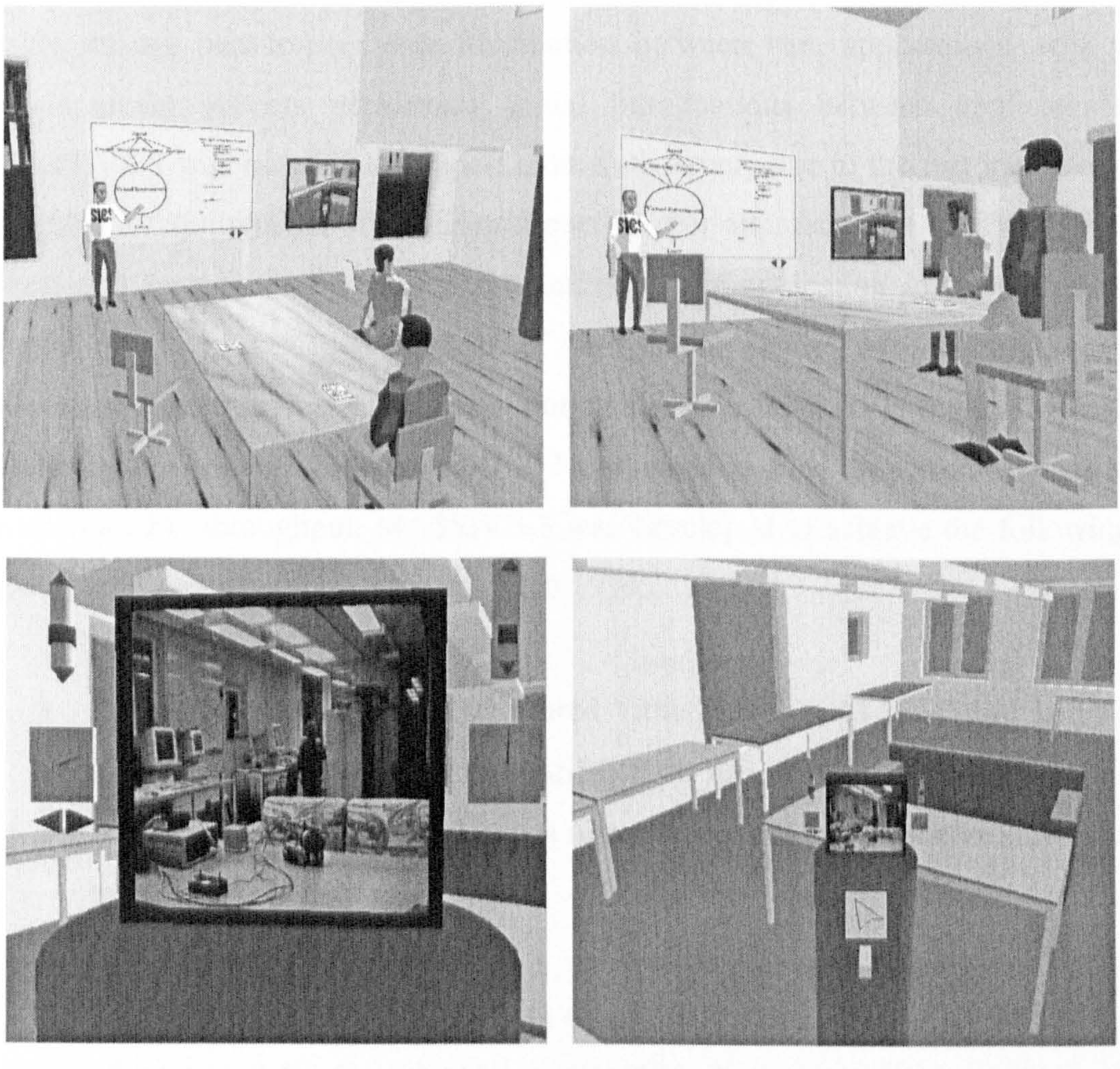


Figure 3-4 Distributed VE networking conference and 3D Interface for robot applications

3.3.4 MASSIVE

The MASSIVE (Model, Architecture and System for Spatial Interaction in Virtual Environments) was developed at the University of Nottingham and has been used to support an extended series of experimental trials of teleconferencing over wide area networks (Greenhalgh and Benford 1995).

The system has three important generations. MASSIVE-1 was a direct and complete implementation of the spatial model of interaction. Its main application domain was teleconferencing and small group meeting. It was implementation using unicast peer-to-peer data distribution between user applications, with a single server process performing initial introductions between applications. MASSIVE-1 was not able to support more than twenty due to the fast increase in bandwidth requirements of the unicast peer-to-peer approach. MASSIVE-2 was developed by using network-support multicast communication in 1996. It also implemented an extended version of the spatial model of interaction and possessed a more sophisticated interaction model than MASSIVE-1. MASSIVE-2 could support up to 20 users on a SGI O2 workstations. This limitation was due to available CPU throughput. MASSIVE-3 was developed to achieve the following goals based on its previous generations in 1999:

- To support hierarchically structured virtual objects. MASSIVE-1 had no support for sub-objects, and MASSIVE- 2 had only limited support; this was because of unresolved issues in reasoning about compound objects in the spatial model of interaction.

- To implement proposals for data consistency and ameliorating the effects of network delay from the University of Reading.

- To provide a route to support heterogeneous computers and networks, including domestic users with modem-based connections.

- To develop a capable and flexible mechanism for interest management (i.e. for structuring virtual worlds and the interaction that occurs within them).

The core part of MASSIVE-3 is a distributed database, which allows application to share descriptions of objects and activities within a 3D audio-graphical virtual world (Greenhalgh et. al. 2000). Each database describes one portion of a virtual world and contains numbers of hierarchically structured data items. A data item can be one of several standard types including a 3D transformation metric describing any combination of rotations, scales, a 3D geometry, a textural and a behaviour defined by a piece of application-specific code. The object's behaviour in the database comprises a name that identifies an embedded class to be instantiated, plus a list of argument values. Each virtual world has a name, a unique system identifier and IP address. Any application that has joined a world can add new objects to it, and change or delete current objects. The change of world is handled by a special event object. The multicasting communication is actually implemented as a client-server system by using unicast network protocols (TCP/IP).

One of the main concerns in MASSIVE system is the interest management (Greenhalgh and Benford 1995). The technique is to divide a virtual world and its contents into appropriately sized "chunks" of information and communication, and to allow each application to deal with only those chunks that are directly relevant to it. In MASSIVE-3, a hybrid method of combining several interest management techniques is adopted. The method integrates locales, functional and organizational distinctions, and abstractions. Basically, the whole virtual world in MASSIVE-3 is split into several locales, which corresponds to a distinct region of the virtual world such as a room, corridor and so on. Each locale defines its own independent co-ordinate system and there is no single global co-ordinate system for the whole virtual world. Different locales can be linked together by applying a 3D transformation that defines the relationship between the locales' co-ordinate system. A locale is further subdivided into one or more "Aspects", which are

defined by several properties such as some functional classes, organizational scope, cost of execution and so on.

The MASSIVE-3 has been used to create a variety of applications, especially in the area of public participation in online art and performance. The *Mirror* was an experiment between BT Laboratories, Illuminations Television, the BBC and Sony in 1997, which involved public access to series of six virtual worlds on the Internet.

The experiment ran in parallel to the BBC television series — The Net. After each TV program, its viewers were invited to become inhabitants in a virtual world whose design mirrored its theme and where they could engage in various events such as debates between performers (e.g. Kevin and David) and playing with interactive objects (e.g., a bouncy castle). Thus, viewing and inhabiting were separated in time. The software used was Sony's Community Place that provided for text and graphical communication between inhabitants and which supported access using a standard PC and modem. Over 2300 people registered to become inhabitants of The Mirror spending over 4400 hours logged on to the server throughout the series. Figure 3-5 shows a scene from Memory in *The Mirror*.

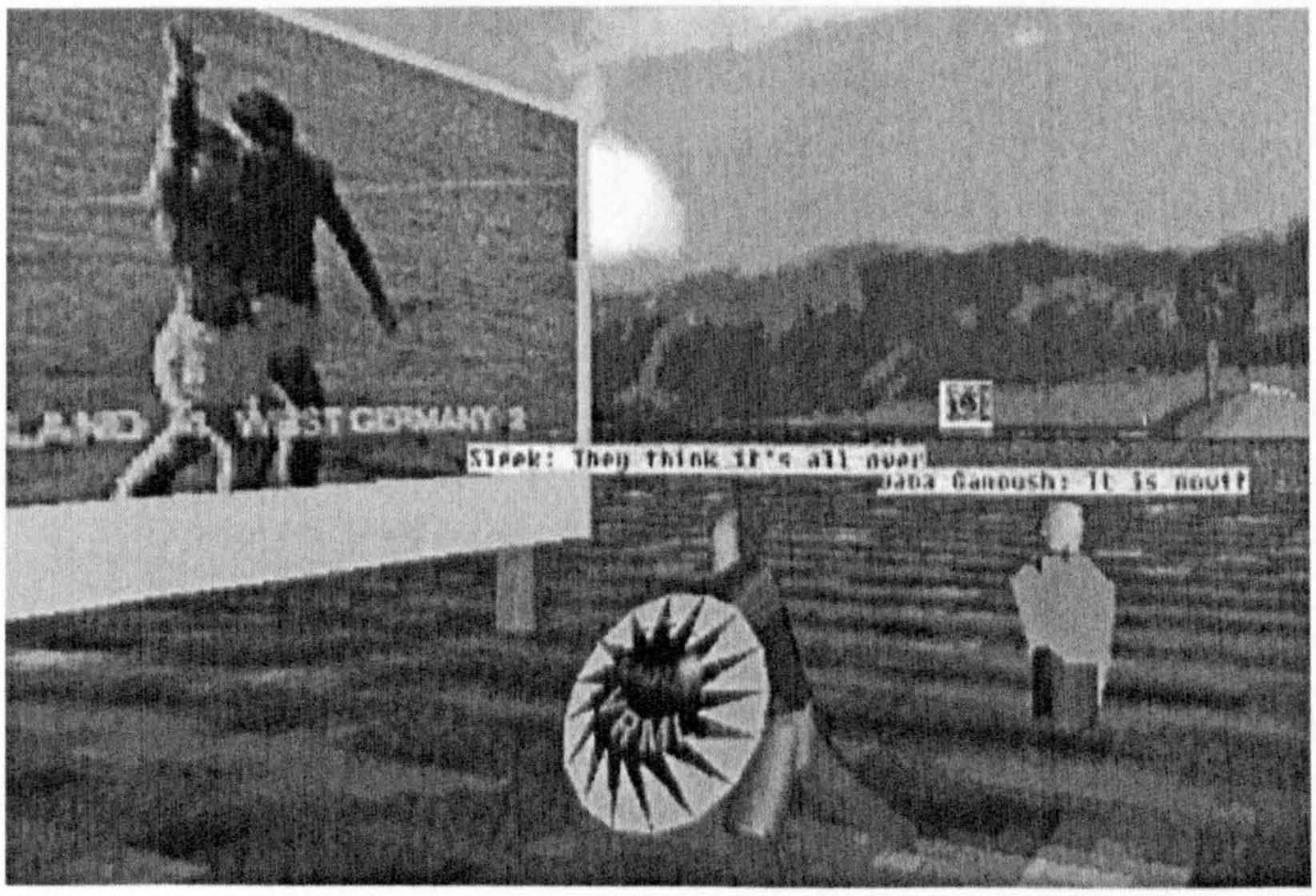


Figure 3-5 The scene of The Mirror

This work has led to the idea of inhabited television as an application area for collaboration Net-VEs. This combines CVEs with broadcast television to create a new entertainment medium in which the public participates in TV shows that are staged within a shared virtual world. The action from these shows is captured by virtual cameras within the world, mixed down and broadcast to more passive viewers as conventional television.

3.3.5 SPLINE

Scalable Platform for Large Interactive Networked Environments (SPLINE) was a collaboration Net-VE system developed by Mitsubishi Electric Research Laboratories (MERL). SPLINE provides a software platform which makes it easy to build virtual worlds where multiple people interact with each other and computer simulations in a 3D visual and audio environment (SPLINE 2002). SPLINE could maintain a distributed, modifiable, and extendable model of a virtual world that is shared between the participants (Waters et. al. 1996).

In particular, SPLINE provides a convenient architecture for implementing multi-user interactive environments. This architecture is centred on a world model that mediates all interaction. Figure 3-6 illustrates several applications interacting through the world models involved the SPLINE architecture.

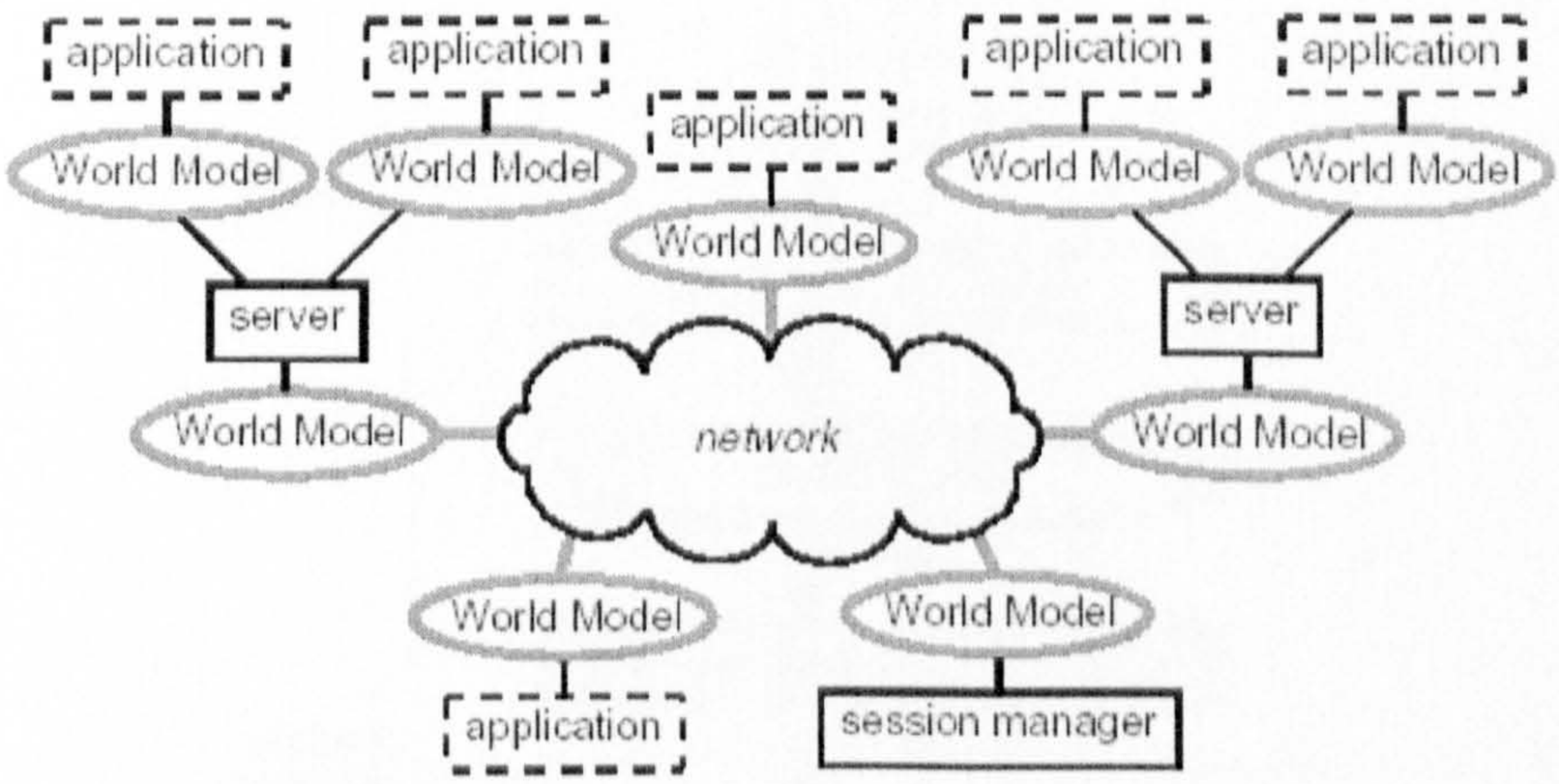


Figure 3-6 Several applications interacting with the SPLINE.

SPLINE applications do not communicate directly with each other, but rather only with the world model. This allows applications to be written without thinking about how communication is achieved. An application does exactly the same things when it is interacting with an application running in shared memory on the same machine as it does when interacting with an application connected via the Internet.

The world model specifies what objects exist in the virtual world. As the virtual world changes second by second, the world model changes. The emphasis in the design of the world model is on the term `database', not `object oriented'. Applications observe the virtual world by retrieving data from the world model. Applications affect the virtual world by adding, removing, and modifying objects in the world model.

Figure 3-7 shows the structure of a SPLINE process. The inter-process communication module provides mechanism to maintain approximate consistency between the world model copies associated with a group of communicating SPLINE, sending messages describing changes in the world model caused by the local application and receiving messages from other SPLINE processes about changes made remotely. The network interface of SPLINE specifies the format of these messages.

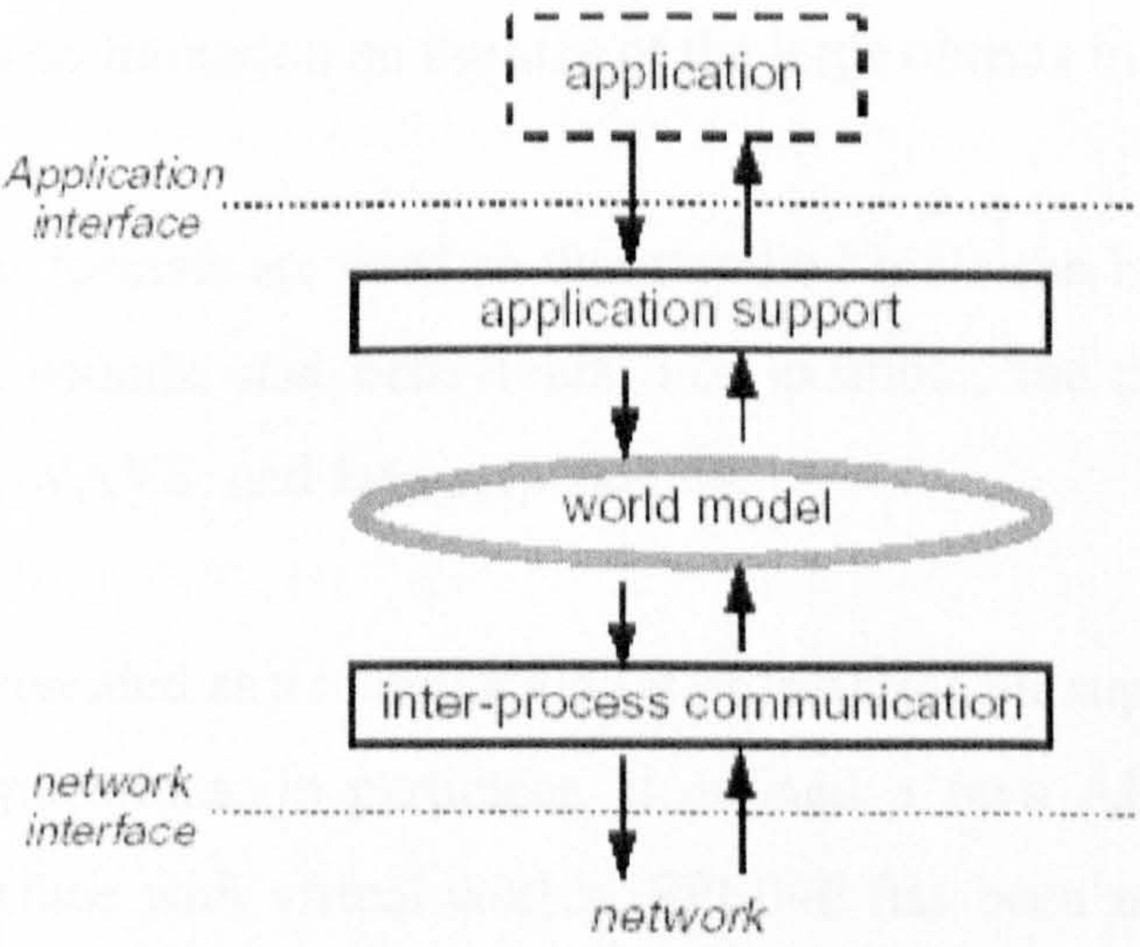


Figure 3-7 The structure of a SPLINE process

SPLINE has the following the main features:

- SPLINE uses various messaging approaches to deal with three types of messages, corresponding to three types of data in the world model: small rapidly changing objects, large slowly changing objects, and continuous streams of data. An important feature of SPLINE is that it includes an efficient scheme for synchronizing these different kinds of data.

- Rather than defining complete virtual worlds, SPLINE defines smaller regions — "locales" that could be joined together to form complete worlds or networks of spaces. Locales could be joined together in very flexible ways.

- SPLINE allows the features of small objects to be changed very rapidly. Messages describing changes in small objects are sent using single User Datagram Protocol (UDP), which allows them to be communicated very rapidly.

- Graphic models, recorded sounds, and behaviours are represented using large objects. These objects are identified by Universal Resource Locators (URLs) and communicated using standard World Wide Web protocols. There is no limitation on the size of the large objects in SPLINE.

- Standard formats are used so that standard tools can be used to create 3D models, sounds, and behaviours. For example, the primary formats are VRML, WAVE, and Java respectively.

SPLINE was presented as a middleware set of facilities for supporting the creation of end-user applications. In particular, it defined a Java API that was used to create and interface with virtual worlds. SPLINE has been used to implement a variety of Net-VEs applications. *Diamond Park* is a social virtual reality system in

which multiple geographically separated users can speak to each other and participate in joint activities. The central theme of the park is cycling. Human visitors to the park are represented by 3D animated avatars and can explore a 3D terrain. In addition to human visitors, the park hosts a number of computer simulations including tour buses and autonomous animated characters.

3.4 The analysis of challenges in Net-VEs

According to the above survey, it is obvious that there are some challenges in a Net-VE because of the involvement of various existing application services such as geometric modelling tools, real time interaction systems, database systems and other transaction systems. This section carries out the analysis of challenges in Net-VEs. These challenges can loosely be divided into three categories, namely content, delivery and architecture (Capps and Stotts 1997) (Benford et al 2001).

3.4.1 Content

Content denotes the objects populating the world, as well as its physical extent, limitations and characteristics of the world space itself. The main problems in Net-VEs can be categorized as:

- **Specify structure and behaviour of world components**

  The virtual world is composed of many different objects that need to be created before any user can interact with it. This process involves several skills such as 3D modelling, programming and graphic art design. Several standards have evolved and are in wide-use including graphics packages such as OpenGL, DirectX and VRML. VRML allows some scripting to give basic behaviour to objects. Emerging standards such as XML, X3D and MPEG4 are trying to improve from the previous standards to allow users create more complex components in Net-VEs.

- **Leverage from semantics, characteristics, ontology of objects and worlds**

  Semantics denotes the behaviour of objects in a VE. Different forms of semantics are distinguished as informal semantics, formal semantics and analysable semantics. Informal semantics are text attachments that human can understand, telling the behaviours, meaning or purpose of an item in VE space. This method will not become practical until the major problems are solved in natural language understanding as no program or autonomous agent will be able to make use of this information. The formal semantics are expressed in some notation that can be processed by algorithm. The behaviour of an object is encoded as a Java/C++ code and is realized by execution. Analysable semantics are formal semantics expressed in a notation. The notation can be operated on by some inference engine allowing autonomous agents to understand the behaviour of objects and make decision based on the percept information.

- **Environments supporting authoring**

  An authoring environment provides systematic support for developing complex worlds. However, VE components are likely to be heterogeneous in various ways such as different execution platform, data format, functionality etc. The idea of creating a middleware as a layer upon different interconnected VE was explored. Use of standard notations such as VRML will alleviate the need of interoperation and composition by eliminating some of the heterogeneity. When all the systems express their models in a common format, no middleware is needed to manage the disparities.

3.4.2 Delivery via the network

The problem of delivery for Net-VEs is how to share the objects among many users in a reliable and scaleable way. Problems can be divided into three folders, namely network bandwidth and delay, scalability and network reliability.

- **Network bandwidth, latency and delay**

  It is believed that network bandwidth poses the primary problem for building large-scale multi-user VEs. For instance, as most of modern Net-VEs systems are normally concerned with managing and rendering large scale 3D environments in which multi-users are capable of interact with each other in real time. Both an over-complex 3D object with huge counts of polygons and very high-resolution image mapped on virtual objects can lead to display latency. The latency can result in the loss of the feeling of immersion. Network bandwidth is limited although there has been rapid growth in recent years, especially for the user, who connects to a Net-VE via a modem connection. In addition to the efforts on optimising the graphics aspect of the Net-VEs system, techniques that are essential to reduce the size of communication about system occurrences are being developed in other distributed-system communities.

  The problem of network delay is a complex mixture of several factors such as network congestion, signal path and processing load through the trip of a network packet. This delay would likely cause the inconsistency among clients regarding to the update of the world in several ways. For instance, an action that is already taken by some participants may not yet have arrived at all points in the world. Efforts on solving this problem fall into two categories, which are an attempt to reduce the source of delay and an attempt to ameliorate its effects.

- **Scalability**

  The requirement to support real-time interaction among large numbers of simultaneous participants distributed over a wide area network makes Net-VE a challenging class of application especially with regard to scale. The scalability of a Net-VE system can refer to the graphical and behavioural complexity of virtual worlds and their contents. Large numbers of active participants generate high volumes of network packets. Server of the network may have to process the data, for instance, in updating a consistent world from many update messages from multi-users. Furthermore, even if the information can be delivered to participants, their local computer must be able to process it and render the shared virtual world while maintaining a sufficiently rapid response to the participants' movements and other actions. Interest management was proposed to address this problem during the last decade. Generally, participants in a world provide some description about what information they are not interested in and those unwanted messages are culled via a system topology. The simplest solution to this problem is the method called regional interest. In such a system, a participant is only aware of other participants within a certain distance or within a predefined grid area of the world. An approach of this inter-user visibility culling is to use the pre-computed hierarchical visibility. In addition, message management need not only be spatially based. For instance, a client with no capacity to modify the world would be uninterested in messages relating to objects' properties such as appearance. Functional groups can be developed for a similar manner of interest management. For example, a participant is only wishing delivery of updates regarding members of a certain group of other participants.

- **Network reliability**

  Network reliability is the rate at which messages are lost, or misdirected. A Net-VE system should be able to withstand the difficulties associated with lost packets. Researches have been conducted on proposing more reliable network protocols. Network link failures also would result in the network reliability problem. A large-scale system should be able to survive such losses without a subsequent loss of presence for participants. For instance, if a participant in a simulation is lost due to a machine or link failure, the system should be dynamically re-configurable such that an available host can begin to simulate that participant.

3.4.3 Network architecture

A Net-VE system should possess good network topology in order to support varying numbers of geographically distributed users and keep participants up to date with the changes in the world. Several distributed architectures have been proposed, which are outlined as follows:

- **Client/server**

  In this architecture, each participant's application communicates only with a common server program that is responsible for passing messages on to other clients as appropriate. This approach is also the standard for public Internet Net-VEs, as the server can tailor its communication to match the network and machine capabilities of each client.

- **Peer-peer unicast**

  Each individual client program sends information directly to other client programs as appropriate. This is typically the most bandwidth-intensive of

the other approaches, but it avoids placing additional load on particular server machines and has less network delays.

- **Peer-to-peer multicast**

  The information is sent simultaneously and directly to many other client programs. It normally uses bandwidth-efficient network mechanism such as IP multicasts. However, multicasting is not currently available on all networks or operating systems. As a result, some systems now include application-specific multicast bridging and proxying servers, which simplify use over wide-area and non-multicast networks. A key area of research is exploring new methods of combining these architectures to efficiently support a range of applications and media over mixed infrastructures.

In addition, Net-VEs raise challenging research questions on interaction issues such as cognitive and motivational. Such higher-level interaction still has not been achieved in Net-VEs although researchers and practitioners have made effort underway to address the natural way of rich interaction issues (Manninen 2000).

Due to the complexity of Net-VEs systems, it has been difficult to design and implement a Net-VE system correctly and effectively. These challenges in the design and implementation process of a Net-VE system depend on the development of computer graphics and network technologies.

3.5 Propose a cost-effective Web-based VR approach

According to the survey of Net-VEs described in Section 3.3, it is obvious that Net-VEs are also a powerful solution for collaborative design and manufacturing applications, which demand not only the sharing of data at distributed locations linked by network, but also the creation of remote 3D realistic representation and real time interaction. However, most previous Net-VEs systems could only be used on the high-bandwidth local area networks like the Ethernet due to the limitation capacity of the networks. With the dramatic change of network technology over the past few years and the increasing speed of the Internet, the potential growth of Net-VEs has become beneficial for Web-based users such as the SMEs, education and customer service applications, and entertainment. Using standard Web browsers as an execution engine for Net-VEs, the Internet has increasingly become the most common location for Net-VEs. This section proposes a cost-effective approach to applying Web-based VR to support design and manufacturing applications via the Internet. The distributed VR system in a Net-VE by using the WWW has been designed by the author for interactive applications in design and manufacturing in order to benefit multiple users, especially SMEs for collaborative e-service.

3.5.1 The architecture design

Since the Internet has become a feasible network for Net-VEs deployment, Net-VEs should have the ability to enter the Web browser. As a developing standard for describing interactive 3D scenes delivered across the Internet, the Virtual Reality Modelling Language (VRML) as the only international standard (ISO/IEC 14772) for Web 3D allows users to navigate and interact with 3D objects in a virtual environment on the Internet. The VRML world browser (e.g. the Cosmo Player) provides an interactive interface that lets multiple users access the Web-based virtual world. In this research project, a cost-effective approach has been proposed by the author to create distributed VR systems for interactive applications in design and manufacture via the Internet for e-service. Figure 3-8

illustrates the framework of the proposed Web-based VR system and outlines the connectivity between its main components.

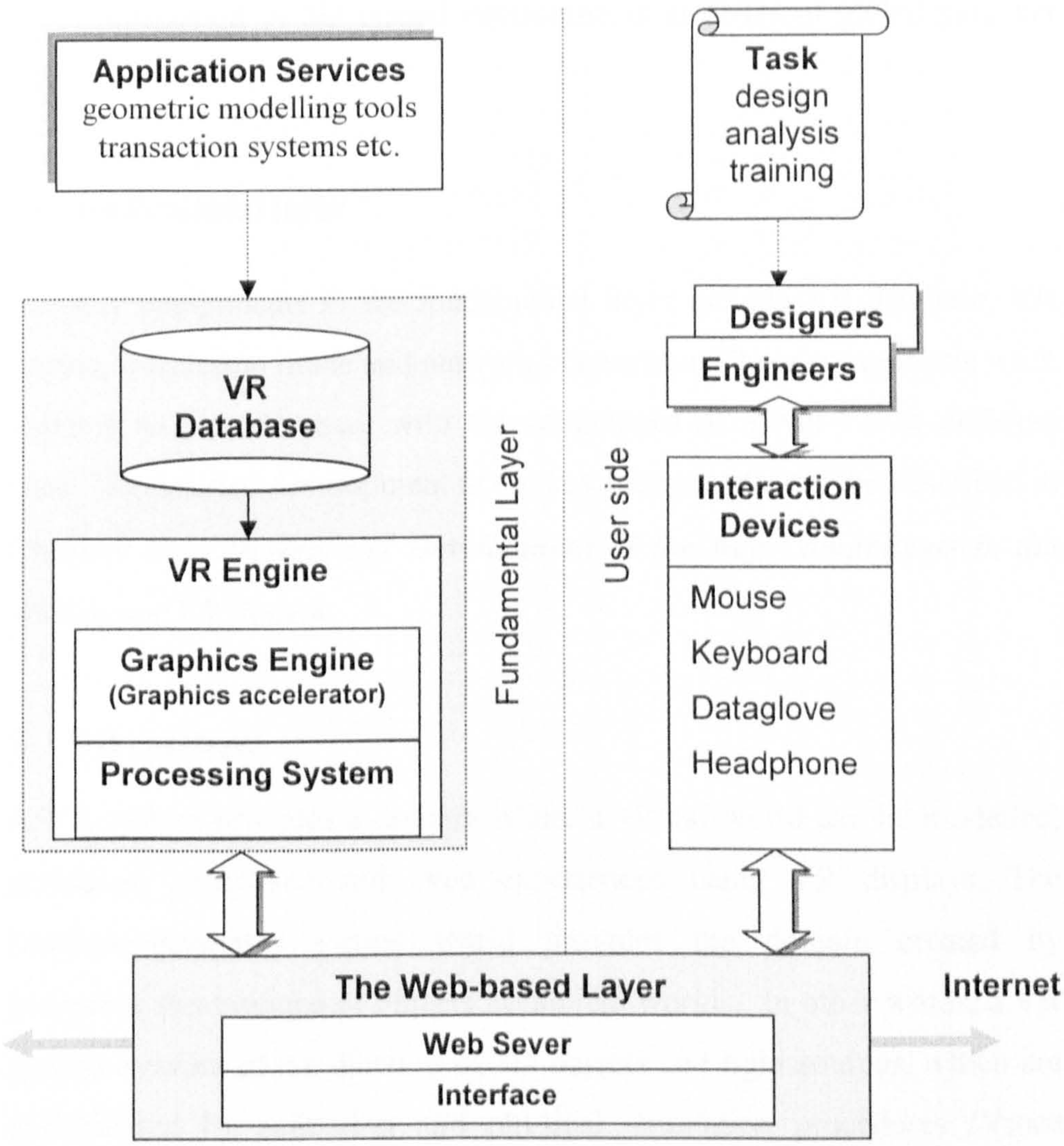


Figure 3-8 The framework of a Web-based VR approach (Proposed by the author)

The system consists of a fundamental layer and a Web-based layer for the Internet access. The fundamental layer is a generic VR system. It supports not only the common monitors but also stereoscopic large-screen projections. It allows engineers to immerse in a virtual design and manufacturing environment by using a mouse, keyboard and even Dataglove. The Web-based layer relies on the Internet as the primary medium to create a sharing environment for its

distribution. Using standard Web browsers as an execution engine for the simulation system, it enables multiple users such as engineers, machinists and trainees to remotely access, navigate, and interact with design and manufacturing applications represented in 3D virtual environments at different global sites via the Internet.

- **The fundamental layer**

The key components in the fundamental layer include VR database, VR engine, interaction mode and network connection. These components work together to provide users with the advantages of a Net-VE at different sites. The detailed development of the key components will be described in **Chapter 4 — *Design and Development of the Key Components in the Web-based VR System*.**

1) VR database

A VR system provides a domain where a virtual world can be modelled, simulated, visualised and even experienced using VR displays. The computer-generated virtual world provides the domain created by geometric descriptions of objects in the real worlds. In other words, a VR system consists of a collection of 3D objects and light sources, which are manipulated by animation and physical simulation procedures (Vince 1995). Thus, the synthetic virtual environments need to be modelled off-line using dedicated software libraries and databases. Such VR database describing virtual objects is stored on disk and loaded into system memory whenever required. The more complex scenarios are needed in a VR system, the larger databases have to be created. For a very large scene, only that portion potentially visible to the user is loaded – the rest resides on disk and is loaded as and when needed. In order to ensure image generation in real time, it is necessary that the size of the active part of database be kept to a minimum.

2) VR engine

The VR engine consists of a graphics engine and a processing system. The graphics engine is responsible for mapping, creating texture, lighting, rendering, and displaying the realistic images in real time. The processing system has the ability to process huge amount of data calculation associated with object dynamics and physical constraints, collision detection, Level of Details (LOD), etc. It is also responsible for coordinating various I/O controls supplied by users and communication among multiple users. Therefore, the VR engine can be regarded as the infrastructure of a Net-VE. Normally, the VR engine can be implemented on multiprocessor graphics workstations. With the advent of graphics accelerators, sufficient graphics capabilities have become available on standard PCs.

3) User interaction mode and devices

Interactions between users and the VR engine are achieved by using various I/O and communication devices. By using the most common input devices, such as mouse and keyboard, a VR system allows the user to manipulate 3D virtual objects in real time and to navigate through the environment. In addition, there is a wide range of more effective devices available to support a VR task. Typically this includes 3D trackers, headphones, sensing gloves, haptic devices, and even software interface and virtual sensors defined in the virtual environment.

4) Network communication

Network plays an essential role in a Net-VE. It distinguishes a Net-VE from a standard VR. Multiple users in a Net-VE rely on the network to share and exchange information for collaborative working. At the same time, the network also supports audio and video communication among users.

- **The Web-based layer**

  The Web-based layer consists of a Web server and an interface for the Internet access. Figure 3-9 illustrates the design of the Web-based layer. The server includes a virtual design and manufacturing application environment along with the databases required. All 3D modelling data is converted into VRML 2.0. The database includes a virtual machine model library along with engineering and manufacturing data.

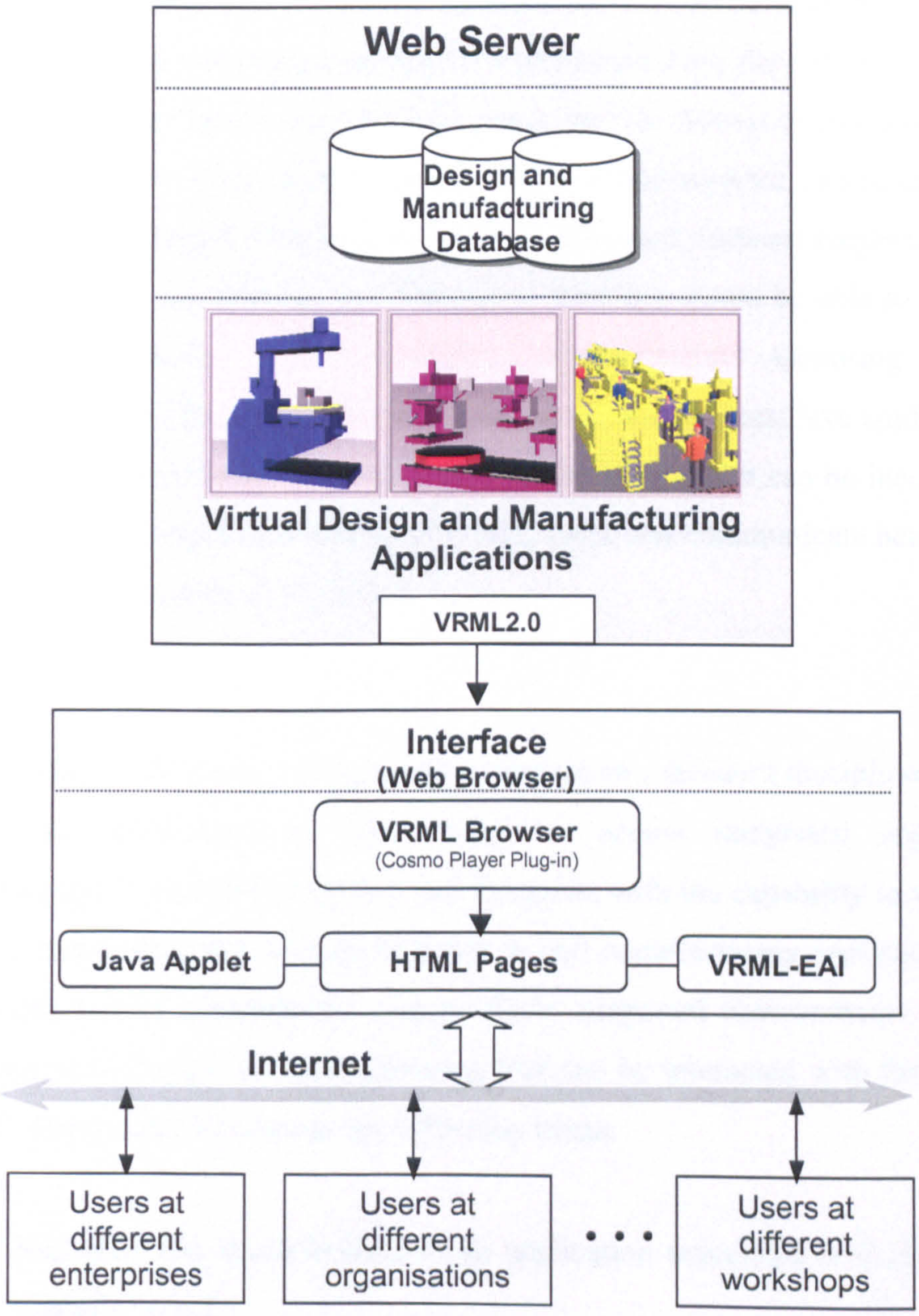


Figure 3-9 The Web-based layer design (by the author)

The Virtual Reality Modelling Language (VRML) as an International Standard (ISO/IEC 14772) provides a tool for the description of interactive 3D scenes delivered across the Internet (Carey and Bell 1997). In the proposed Web-based VR approach, VRML is adopted as the visualisation integration technology to present design and manufacturing applications on the Web. It represents static and animated dynamic 3D and multimedia objects with hyperlinks to other media such as text, images, sounds, and videos. The characteristic of VRML is consistent with that of Hypertext Markup Language (HTML). Therefore, the 3D scenes of an application in design and manufacturing can be transformed from the fundamental layer to the Web-based layer by converting the file format of 3D models into VRML 2.0. The Cosmo Player, a VRML world browser, can be embedded into the normal Web browser such as Microsoft Internet Explorer (IE) or Netscape. In addition, the Web-based interface would be able to combine VRML browser with Java applet and the External Authoring Interface (EAI). EAI has the mechanism to communicate between Java applets and a VRML scene graph. As an input tool, the Java applet can be incorporated to execute the EAI to update a VRML scene and communicate between the client and server (EAI 2001).

The proposed Web-based VR approach integrates two different disciplines: virtual reality and networking to allow users to access integrated engineering environments. It provides engineers and designers with the capability to visualise, explore, manipulate and interact with design and manufacturing applications in a distributed virtual environment system. Such integrated environments simulate applications in design and manufacturing that can be interacted with through the WWW. The system focuses on the following items:

- Represent the characteristics of an application associated with design and manufacturing.
- Simulate a dynamic process in a computer generated 3D virtual space.

- Share/Interact with 3D data and information in real-time.
- Reduce the risk of setting up an actual system before evaluation.

By reducing costs and cycle time, the distributed VR system is able to enhance and even accelerate the major activities of manufacturing engineering, including: manufacturing process concept development and simulation, optimising assembly lines and workshops design, integrating labour and equipment, etc.

3.5.2 Prototyping potential applications

Since the design and manufacturing activities are diverse, this section discusses three potential applications which can be prototyped based on the above approach proposed by the author. They respectively concern three applications: machining, process planning and factory layout.

- **Machining**

  This application facilities the visualisation and analysis functionality of a machine. It emulates components, tools, controllers and workpiece along with a machining process in a 3D visual way. It enables users to preview and explore a new or valuable machine in virtual environments before its purchase. It can be used to train engineers and machinists on the processes of a machine set-up, tool change and production programs. The application focuses on the representation of 3D dynamics machines in real time. The different modelling methods are integrated for this purpose. At first, since Deneb's Virtual NC, a commercial virtual manufacturing software package, provides libraries of 3D virtual machines, it is the choice for anyone wanting a specific virtual machines library. In order to reuse directly the machine models in a virtual world generated by VRML, the model files from Virtual NC should be exported as IGES format. Then, they are converted into VRML 2.0 by a 3D data translation tool. Most common machine models like mills and lathes are achieved by this

method. On the other hand, a hybrid method is proposed by the author to combine the capabilities of VRML geometric modelling with some advantages of commercial modelling packages such as Kinetix' 3D Studio Max. It can be used to represent some new types of machines such as the latest Computer-Numerical-Control (CNC) machines.

- **Process Planning**

  The application is designed to represent a process flow in a virtual workshop. It aids process planning by process flow simulation and analysis. It is a useful tool for reducing risk and shortening the cycle of product development before establishing a real process flow. The application emphasises the creation of animations for a manufacturing process. The basic solution is based on the dynamic mechanisms of VRML that combine Interpolator nodes with Sensor nodes to generate simple animation calculations. Arbitrary behaviours can be simulated once script nodes are bound up with the Java and JavaScript languages. In addition, since Deneb's QUEST as a professional manufacturing simulation package can directly emulate real-world process behaviours, its simulation capabilities (Deneb Corp. 1995) can be coupled with VRML worlds via a translator (Wang, 2000) provided by National Institute of Standards and Technology (NIST), USA.

- **Factory layout**

  The application allows users to design work places and equipment layout prior to the start of production. An optimal factory layout design according to the rules of ergonomics could maximize the efficiency of workers and achieve the full performance of equipment. The prototype simulator enables users to access a virtual factory and provides the ability to layout the virtual machines. It allows users to select and add new machines into the original world and rearrange them by moving, scaling and rotation. The application highlights the reuse of 3D models and the integration of various virtual machines in a VRML world. VRML files exist in a parent-

child hierarchical structure. Models are subparts of the world. Such a structure makes it easy to create large worlds or complicated scenarios, which reuse subparts or inherit attributes and behaviours from subparts. As a result, not only a machine model but also its components can easily be reused in various manufacturing applications.

Due to the time limitation of this project, it is impractical to develop all of the above applications. Thus, a prototype simulation system— VRML-based Factory Layout Simulator (VFLS) as a demonstrator has been developed and implemented for interactive factory layout applications via the Internet on basis of the proposed Web-based VR approach. Its specific implementation will be described in **Chapter 5-** ***Implementation***.

3.6 Conclusions

The chapter firstly presented the concept and features of Net-VEs. It carried out a survey of Net-VE systems and then analysed the major challenges in Net-VEs. Further, a cost-effective approach has been proposed by the author to apply Web-based VR to support design and manufacturing applications via the Internet. The distributed VR system in a Net-VE using the WWW is designed by the author for interactive applications in design and manufacturing in order to benefit multiple users (e.g. SMEs) for collaborative e-service. The detailed architecture design of the Web-based VR system has been presented here. In addition, it discussed potential applications that can be prototyped based on the proposed approach. The next chapter will describe the underlying design and development of key system components — VR database, VR engine, and Network communication in the proposed distributed Web-based VR system.

Chapter 4

Design and Development of the Key Components in the Web-based VR System*

4.1 Introduction

This chapter describes the underlying design and development of the key components — VR database, VR engine, and network communication for the proposed Web-based VR system in Net-VEs to support interactive applications in design and manufacturing.

At first, this chapter describes 3D modelling in the creation of VR database. Section 4.2 exploits the integration of major 3D modelling techniques and the specific geometric modelling capabilities and constraints in VRML2.0. Then, a hybrid method is designed by the author through combining the features of VRML with some advantages of commercial CAD/CAM modelling packages (e.g. Deneb's Virtual NC and QUEST) to develop 3D models of facilities in the proposed Web-based VR system.

Secondly, the chapter presents VR engines for interactive simulation. The performance of the VR engine directly influences the capability of real-time display and interactive simulation in the distributed Web-based VR applications

* The main content of this chapter has been published in the proceedings of **13th European Simulation Symposium: Simulation in Industry**, Oct. 18-19, 2001 at Marseille, France.

that the project is targeting. Section 4.3 proposes a generic framework of VR engine and the model of interactive simulation. It then presents how the capabilities of interactive simulations in VRML are used to develop interactive dynamic behaviour simulation in the proposed Web-based VR system.

Finally, the chapter discusses network communication. Since Web-based VR applications will involve multiple users connected through the Internet in a distributed environment, they require considerable network bandwidth and performance for communication. Section 4.4 firstly describes the networking fundamental issues involved in the design of Web-based VR systems. Then, it examines the Internet protocols — IP unicasting and IP multicasting based on different network architectures. In particular, this section evaluates network protocols by comparison of their advantages and limitations in order to select the most appropriate protocol for different types of Net-VE architecture design. Furthermore, it suggests a hybrid method of reliable communication by using multiple protocols for the proposed system in Net-VEs.

4.2 3D modelling in the creation of VR database

Since a generic VR system consists of a database of objects and light sources which are manipulated by animation and physical simulation procedures, the geometric modelling plays an important role in the development of VR database. The first step of the creation of a virtual environment is to represent 3D objects in terms of geometric modelling in the virtual scene. A wealth of professional modelling systems has been developed such as 3D Studio Max, Auto CAD, Pro/Engineer, Unigraphics etc. This section firstly exploits the integration of major 3D modelling techniques used in these professional modelling systems and the hierarchical structure of a Scene Graph in virtual environments. It then explores the specific geometric modelling capabilities and constraints in VRML2.0. Finally, a hybrid method has been proposed by the author through combining the features of VRML with some advantages of commercial CAD/CAM modelling packages to develop 3D model of facilities in the proposed Web-based VR system.

4.2.1 Exploiting 3D modelling techniques

Modelling is the production of definition of a 3D object in sufficient detail for visualization in a computer graphics system. Modelling techniques can be mainly divided into two categories, namely surface modelling and solid modelling. Whereas solid modelling emphasises the physical properties of objects, surface modelling emphasises the visual aspect.

In addition, modelling is concerned with not only shape, but also form that shows how linked elements are connected. To create a virtual object, the user needs to know its dimensions. The individual elements can be represented separately from the basic geometry. Further, they can be assembled and grouped to compose the more complex object (Halliday and Green 1994). The final numerical description encodes the object's shape and form, and if this data remains constant, the object's geometric integrity is safe (Vince 1995).

4.2.1.1 Surface modelling

Surface modelling contains two main representations methods, namely polygon mesh surface and parametric surface.

- ***Polygon mesh surface***

  Polygon mesh modelling is the most popular 3D modelling method used in present 3D software packages. Fast rendering of a polygon mesh is widely supported by graphics hardware that has become popular on normal computer systems such as PCs. A polygon mesh is a set of edges, vertices, and polygons connected such that each edge is shared by at most two polygons. An edge connects two vertices, and a polygon is a closed sequence of edges. An edge can be shared by two adjacent polygons, and a vertex is shared by at least two edges. There are three polygon-mesh representations, which are explicit, pointers to a vertex list and pointers to

an edge list. Each representation has its advantages and disadvantages and is applied according to different requirements of various graphics systems.

3D models in CAD are highly accurate and model smooth curves with great precision, polygonal modelling applies a technique called polygonal approximation to models consisting of vertices and edges. This technique approximates the curvature of a surface using small flat polygons.

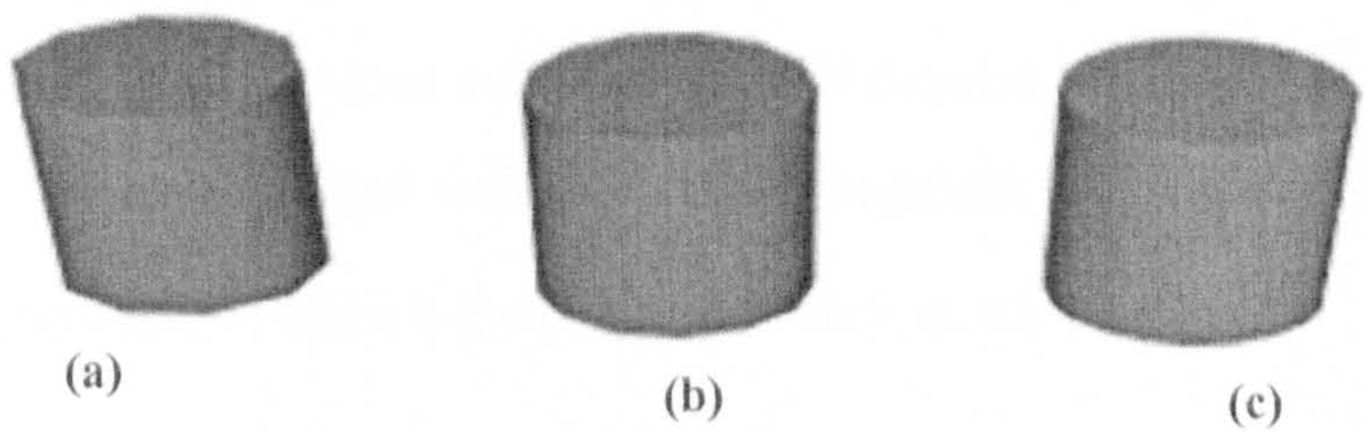


Figure 4-1 Polygon approximation of curve surface. Side segments of the cylinder are 10, 13 and 20 in (a), (b) and (c) respectively, representing smoother curve surface.

As shown in Figure 4-1, the more polygons, the smoother the object is. However, high-count polygon objects present severe problem for real time rendering especially for the Web-based VR system that has bandwidth limitation in this project. Furthermore, maintaining the consistency and smoothness of a complex polygonal mesh is not a trivial task as the mesh data normally are generated from some digitising drawing tools or 3D laser scanner in which case errors are inevitable.

Thus, in this research project, when making polygonal models for interactive Web-based VR applications in design and manufacturing has to produce objects that looks good but use as few polygons as possible ("low polygon count") so that the client computer can download and render them quickly.

- ***Parametric surface***

  As a polygon is a first-degree, piecewise linear approximation to curve surfaces, large counts of polygons are required to represent those smooth curve surfaces. Higher degree mathematical functions are used to alleviate this problem. These functions still only approximate the desired shape but with less storage and easier interactive manipulation than first-degree polygon approximation. Cubic polynomials are most widely used as lower degree polynomials give too little flexibility in controlling the shape of the curve while higher degree polynomials will require too much computation. A cubic polynomial that defines a curve segment $Q_{(t)} = [x(t)y(t)z(t)]$ can be represented by following equations (Foley et. al. 1996):

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x,$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y,$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z,$$

  where $0 \le t \le 1$

  In CAD or common engineering simulation, a curve segment Q_t is defined by constraints on endpoints, tangent vectors, and continuity between curve segments. There are three major types of curves, namely Hermite, Bezier and several kinds of splines (Foley et. al. 1996).

4.2.1.2 Solid modelling

In many simulation applications, it is important to distinguish between the inside, outside and surface of a 3D object and to be able to compute the required properties of the object that depend on this distinction. The need to model objects as solid rather than just a collection of points and lines has resulted in the development of a variety of specialized techniques to represent them. These techniques are outlined as follows:

- ***Primitive Instancing***

  In this method, the modelling system defines a set of primitive 3D solid shapes with several adjustable parameters. The parameterised object is defined by a few high-level parameters. For instance, a gear as shown in Figure 4-2 can be parameterised by its diameter or number of teeth. These features are difficult and time-consuming to be defined using other solid modelling methods such as Boolean combination in Constructive Solid Geometry (CSG).

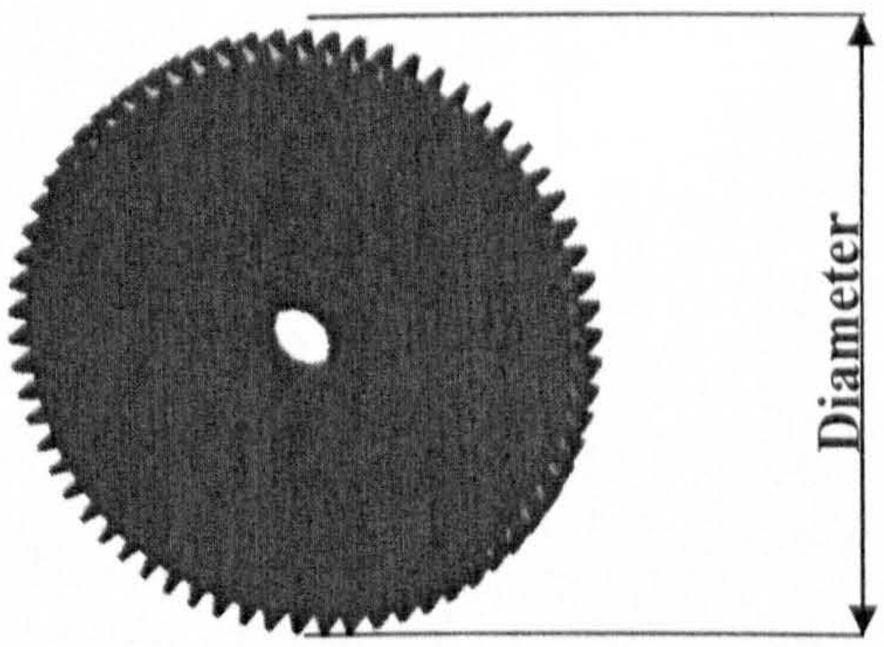


Figure 4-2 Gear model constructed by primitive instancing

- ***Boundary representations (b-reps)***

  This method is the most popular one in all the representations method. An object is described in term of its surface boundaries that are vertices, edges and faces together with its physical properties such as colour, texture etc. Curved surfaces are normally approximated with polygons or represented as surface patches. Two types of information have to be kept about an object, namely, purely geometric information such as the co-ordinates of a vertex and topologic information which details how the geometric entities relate to each other, for example, which two vertices are connected to form an edge of the object.

  To transform the object, only the vertex co-ordinates are required for calculation. The topology among the vertices does not change under

scaling, rotation or translation. If a shaded picture is required then the face list is accessed to obtain information to allow each face to be drawn. Similarly if a wire-frame is required then the edge list is accessed to obtain the start and finish vertices of each edge so the edge can be drawn. Figure 4-3 shows a complex B-reps sharpening tool in wire-frame.

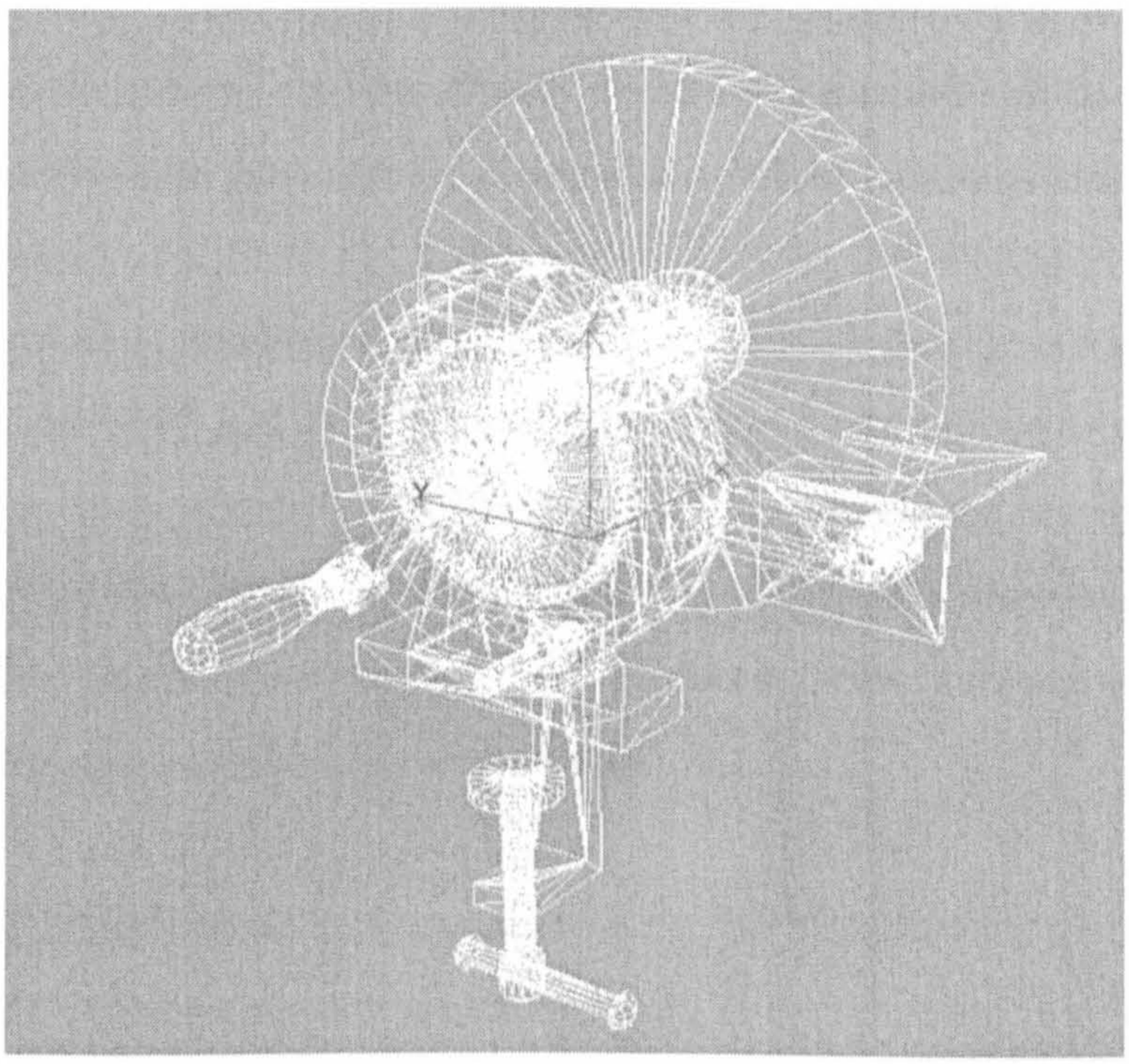


Figure 4-3 A B-reps sharpening tool in wire-frame.
(Object contains 4098 vertices and 8176 faces.)

- ***Sweep representations***

  Polygonal models of objects can be generated by sweeping a cross-section along a curve. The cross-section may be allowed to change in size as it is swept. For example a cylinder can be considered as a circle swept along a line and a torus is a circle swept round a larger radius circle. However, general sweeps are very difficult to model efficiently. For instance, the trajectory and object shape may make the swept object intersect itself. In addition, applying Boolean operations between two swept objects become difficult without firstly converting the objects to other representation. For

example, a union of two simple sweeps is generally not a simple sweep. Thus, modelling systems normally allow the user to produce certain number of objects using sweep but stored the defined object in other representations.

- ***Spatial-partitioning representations***

In this method, a solid object is decomposed into a collection of adjoining, nonintersecting solids primitives such as different-shaped blocks. Several variant of this method have been developed which are listed as follows:

1. ***Cell decomposition*** defines a set of primitive cells that are typically parameterized and often curved. This method can compose complex objects from simple, primitive one in a way of "gluing together". As primitive cells are not allowed to intersect with each other, validation of the object becomes difficult. However, cell decomposition is widely used in finite element analysis.

2. ***Spatial-Occupancy Enumeration*** can be thought as a special case of cell decomposition in which the solid is decomposed into identical cells arranged in a fixed, regular grid. The cell is normally a cube and called voxel. The presence or absence of a single cell at each position in the grid can be altered to represent various objects. The object can be encoded by a unique list of occupied cells. This technique is used in biomedical applications to represent volumetric data. However, it has several limitations. For instance, a partial occupancy cell does not exist, which results in failure to represent some object with a curve surface accurately. Cells can theoretically be made as small as desired to increase the accuracy of the representation.

3. ***Binary Space-Partitioning Tree (BSP)***

BSP tree recursively divides space into pairs of subspaces, each separated by a plane of arbitrary orientation and position (Foley et. al.

1996). The technique was originally introduced to accelerate the process of detecting the visible surfaces in a complex environment (Fuchs et. al. 1980). It was then expanded to represent arbitrary polyhedra (Thibault and Naylor 1987). Each internal node of the *BSP* tree is associated with a plane and has two child pointers, one for each side of the plane. The left child is behind or inside the plane, whereas the right child is in front of or outside the plane. If the half-space on a side of the plane is subdivided further, then its child is the root of a sub-tree. The subdivision continues until the half-space is a leaf which represents a region entirely inside or outside the polyhedron. The *BSP* technique has several variants depending on the dimension, for instance, *Quadtree* and *Octree*.

- ***Constructive Solid Geometry (CSG)***

  A CSG object is constructed via combining simple geometrical primitives such as sphere, cube, cylinder etc. by means of regularized Boolean set operators (Chiyokura 1988). It is stored as a tree with operators at the internal nodes and simple primitives at the leaves. Nodes in a CSG tree can be operator or 3D translation, rotation and scaling. Boolean Operators can be:

  -Union: A + B is the set of vertices that are in A or B.

  -Intersection: A.B is the set of vertices that belong to A and B.

  -Difference: A-B is the set of vertices that belong to A but not B.

  The method of Constructive Solid Geometry arose from the observation that many industrial components derive from combinations of various simple geometric shapes such as spheres, cones, cylinders and rectangular solids. In fact the whole design process often started with some simple block which might have simple shapes cut out of it, perhaps other shapes

added on etc. in producing the final design. For example see the simple solid constructed in Figure 4-4:

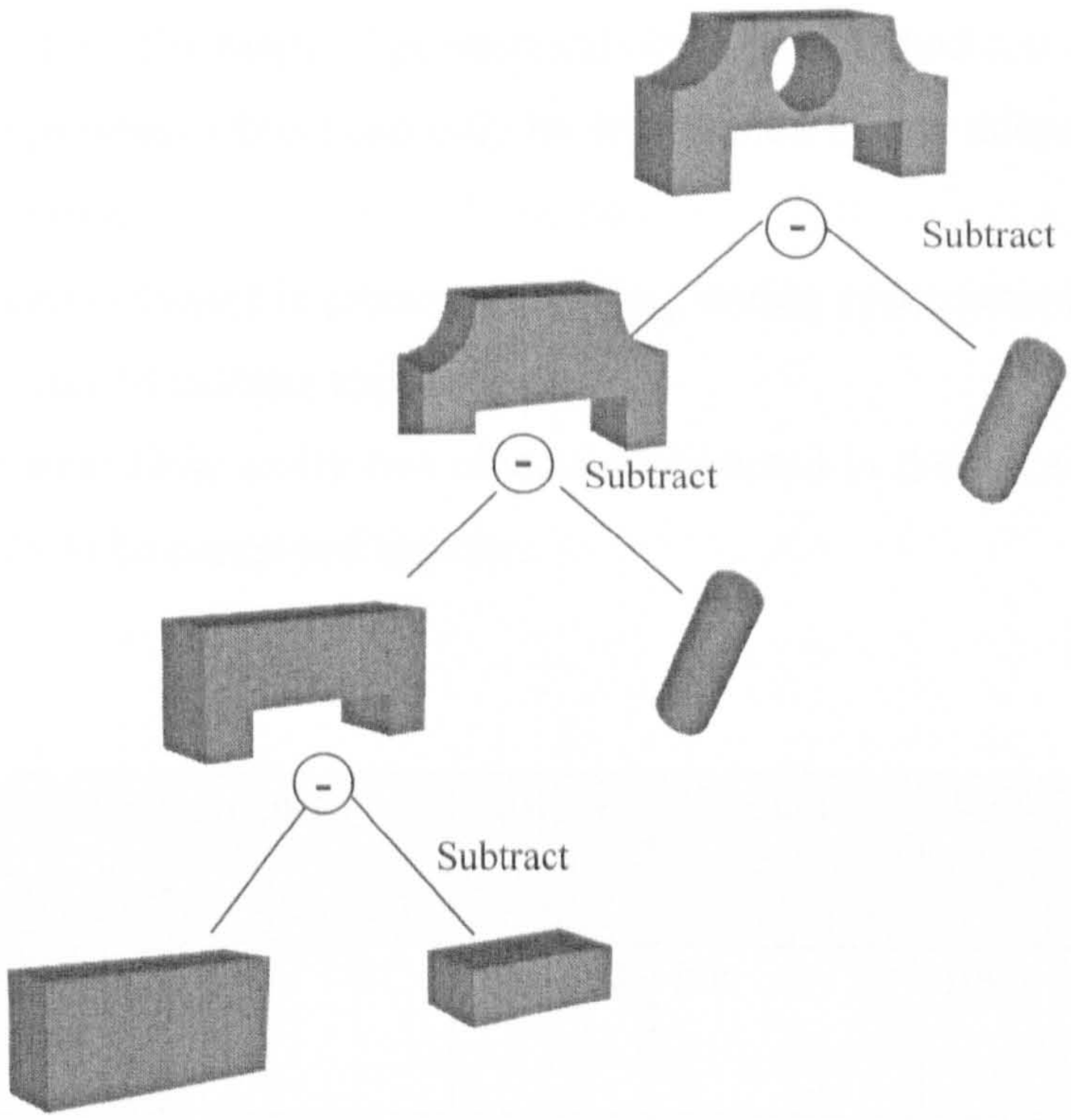


Figure 4-4 A solid object constructed using CSG

CSG is efficient for describing machined objects provided that the primitive objects library is rich. Furthermore, it is able to edit models by deleting, adding, replacing, and modifying sub-trees effectively, which make it become one of the dominant solid modelling representation methods.

- ***Comparison of solid representations***

  Table 4-1shows the comparison of different solid representation methods:

  a) **Accuracy:** Object is represented in an accurate way or by approximation.
  b) **Domain:** The range of geometrical object the method can represent.
  c) **Uniqueness:** Object can only be represented in one unique procedure or several.
  d) **Validity:** Object is prone to go wrong during construction or not, and is it easy to validate the problem.
  e) **Closure:** How easily two objects constructed in the same method are easily to be composed together.

Solid Rep. Method	Accuracy	Domain	Uniqueness	Validity	Closure
Primitive Instancing	Yes	Limited	No	Easy	No
Sweep	Yes	Limited	No	Easy	No
B-reps	Approx.	Wide	No	Difficult	Yes
Spatial partitioning	Approx.	Wide	Yes	No need	Yes
CSG	Yes	Limited	No	Easy	Yes

Table 4-1 The comparison of different solid representation methods

It is noted that most of the modelling systems use multiple representations as some operations are more efficient with one representation that with another. In addition, different representation methods can be converted to each other using the approximation method.

4.2.2 Hierarchical scene graph in virtual environments

In a virtual environment, a complex scene is made up of a number of 3D objects which are based on the hierarchical scene graph. The 3D objects are positioned in a scene by applying various transformations such as translation, scaling, or rotation. A Scene Graph (or object hierarchy) is a tree data structure that is used to store the elements of a 3D scene. The Scene Graph consists of nodes. In addition to shape nodes, and nodes such as lights and cameras, the scene graph can contain grouping nodes with children. Transformation grouping nodes enable objects to be grouped together and transformed (translated, rotated, scaled) as one unit relative to its parent node. Each transform group has its own local coordinate system. Using a scene graph, the user can control the position of objects at different levels.

This is very important as the structure of the Scene Graph, and the manner in which complex objects are split up into parts defines the limits of what is possible if an object is to be animated or made interactive. The structure of the Scene Graph is also important from an efficient rendering perspective as the hierarchy can affect run-time performance. Hierarchical modelling in the Scene Graph builds up complex models by initially starting with some basic simple objects which are combined using appropriate modelling transformations to define more complex objects. In turn these more complex objects can be combined to make still more complex objects. For example a workshop that was modelled by the author as an example of hierarchical scene graph as shown in Figure 4-5.

This workshop could actually be represented entirely by starting with a square and then building up more complex objects such as `a garage', a `window', a `door' etc. This process is illustrated below using a tree structure of the Scene Graph.

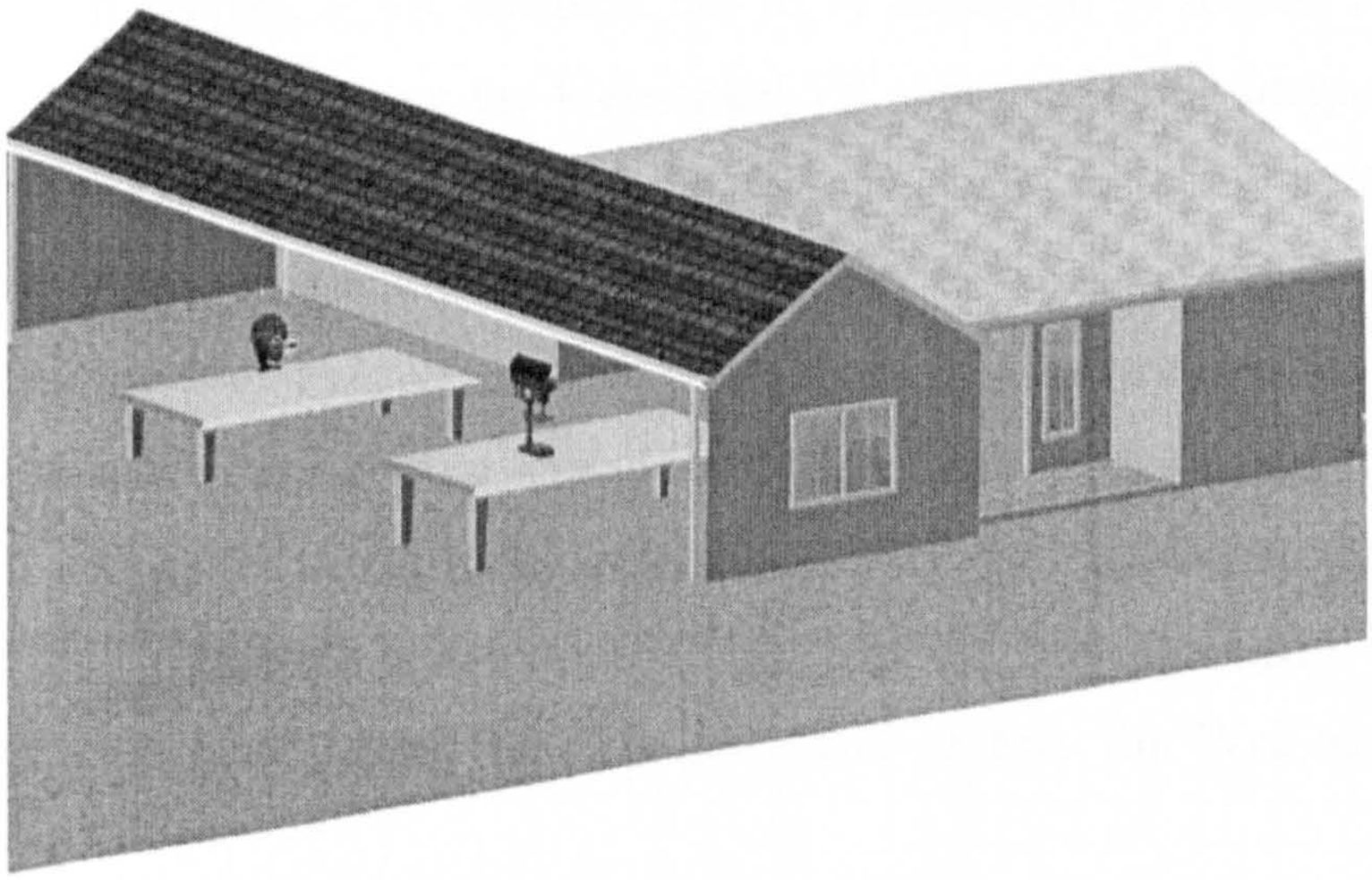


Figure 4-5 The scene graph example – modelling a workshop

In this tree type structure, a transformation is associated with each link. For example table1 is constructed based on its preceding transformation matrices in the higher hierarchy. The transformation matrix of the table1 is calculated by accumulating all these matrices as follows:

$T_b * T_{h1} * T_{t1}$

When building hierarchical models the transformation matrix is saved at each downward step in the traversal of the structure and restored again before traversing upwards.

4.2.3 Examining geometric modelling capabilities in VRML

In this research project, a VR database has to be modelled at several different levels of detail in order to keep the Web-based VR system with an adequate real-time operating and interactive capacity since the proposed system is designed to be delivered across on the Internet. It is necessary to consider minimizing the number of polygons in a virtual scene to make the virtual environment acceptable for different machines from a cheap PC to a high performance graphics workstation and different networks from the dial-up modem connection to the high speed and wide band connection. VRML provides the capabilities of cost-effective geometric modelling for low polygons objects for Web-based VR system.

In VRML 2.0, the ***Shape*** node is defined to perform geometric modelling. It associates a geometry node with appearance nodes including material and texture property. ***Shape*** nodes must be part of the transformation hierarchy to have any visible result, and the transformation hierarchy must contain ***Shape*** nodes for any geometry to be visible. The capabilities of geometric modelling are illustrated as follows.

- The valid geometry nodes: ***Box***, ***Cone***, ***Cylinder***, ***Sphere***, ***ElevationGrid***, ***Extrusion***, ***IndexedFaceSet***, ***IndexedLineSet***, ***PointSet*** and ***Text***.
- A ***Shape*** node contains exactly one geometry node in its geometry field.
- Several geometry nodes contain Coordinate, Colour, Normal, and TextureCoordinate as geometric property nodes which could be shared between different geometry nodes.
- ***Appearance*** nodes are able to describe the appearance properties such as material and texture to be applied to the Shape's geometry.
- Any of the three texture node types (ImageTexture, PixelTexture, or MovieTexture) can be used with any of the geometry nodes.

This following section specifies the VRML nodes for geometric modelling which can be used in the creation of a VR database in the Web-based VR system. The detail definition of the syntax and semantics of some nodes are described here.

- **Shape node:**

  The ***Shape*** node has two fields, appearance and geometry, which are used to create rendered objects in the world. The appearance field contains an ***Appearance*** node that specifies the visual attributes (e.g., material and texture) to be applied to the geometry. The geometry field contains a geometry node. The specified geometry node is rendered with the specified appearance nodes applied.

Shape { exposedField SFNode **appearance**
exposedField SFNode **geometry** }

- **Appearance node:**

  The ***Appearance*** node specifies the visual properties of geometry by defining the Material and texture nodes.

Appearance { exposedField SFNode **material**
exposedField SFNode **texture**
exposedField SFNode **textureTransform** }

- **Geometry nodes**

  The following geometry nodes often are used to construct 3D objects in VRML worlds.

1. **Box :** The ***Box*** node specifies a rectangular parallelepiped box centred at (0, 0, 0) in the local coordinate system and aligned with the local coordinate axes. Figure 4-6 illustrates the Box node.

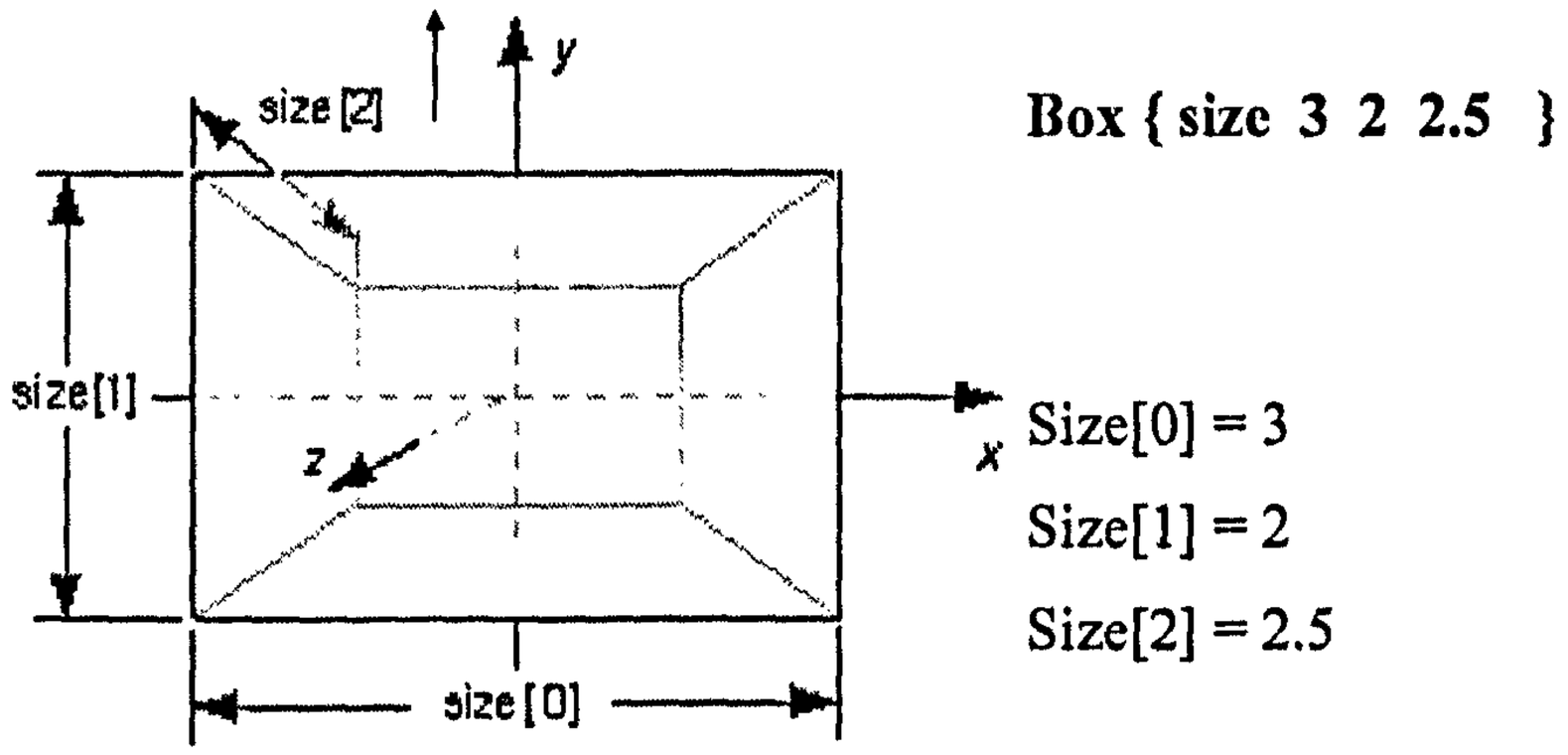


Figure 4-6 The Box node.

2. **Cone :** The ***Cone*** node specifies a cone which is centred in the local coordinate system and whose central axis is aligned with the local Y-axis. The Radius field specifies the radius of the cone's base, and the height field specifies the height of the cone from the centre of the base to the apex. Figure 4-7 illustrates the Cone node.

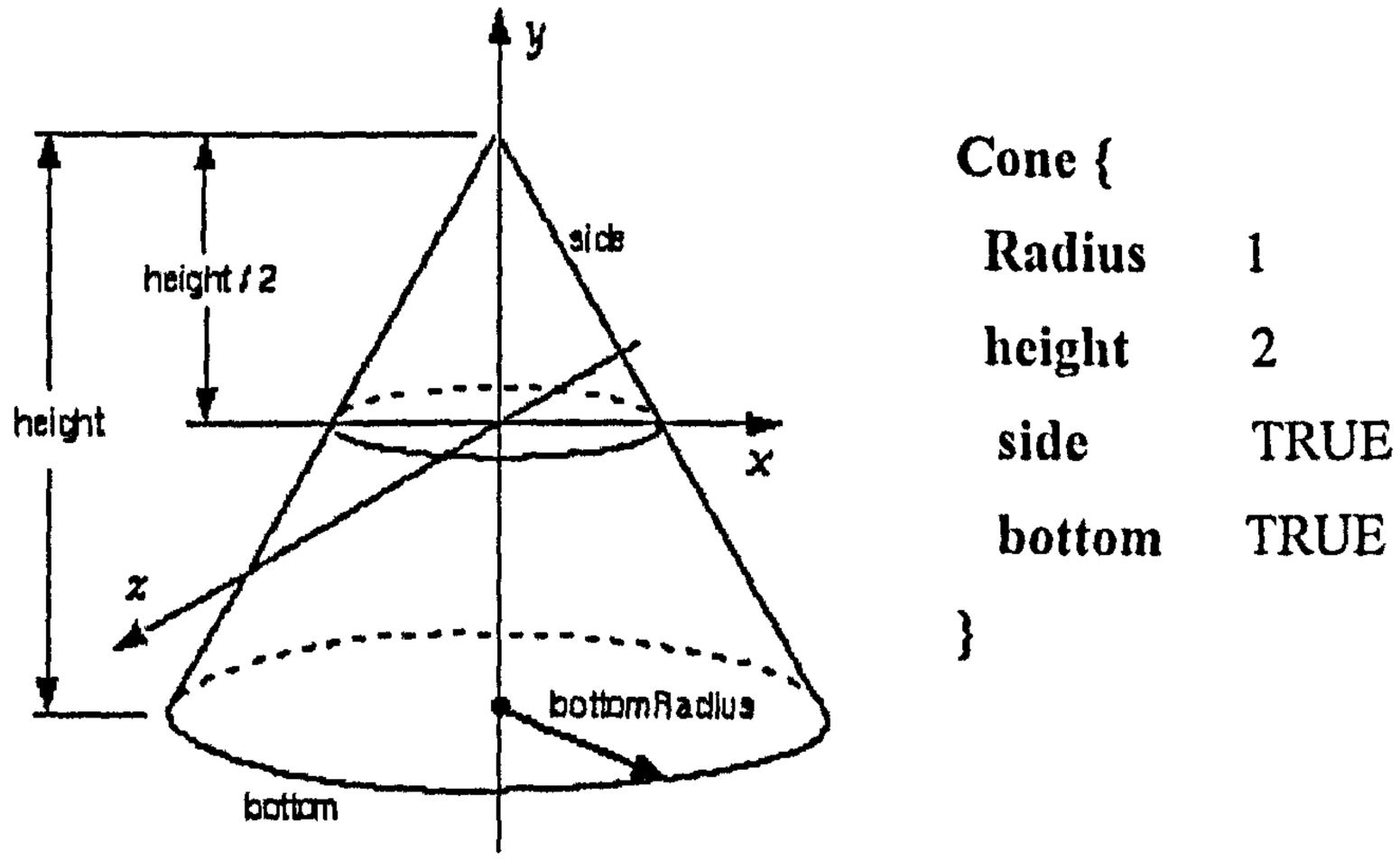
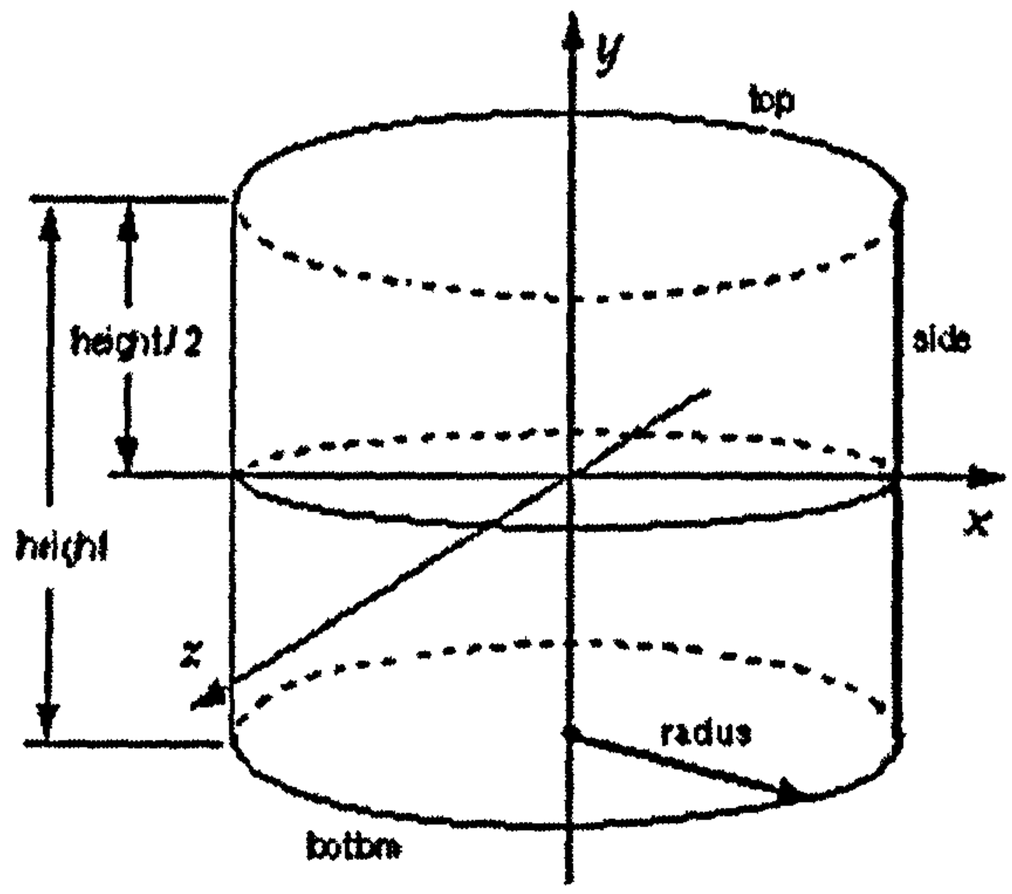


Figure 4-7 the Cone node.

3. **Cylinder :** The ***Cylinder*** node specifies a capped cylinder centred at (0,0,0) in the local coordinate system and with a central axis oriented along the local Y-axis. The radius field specifies the radius of the

cylinder and the height field specifies the height of the cylinder along the central axis. Figure 4-8 illustrates the Cylinder node.



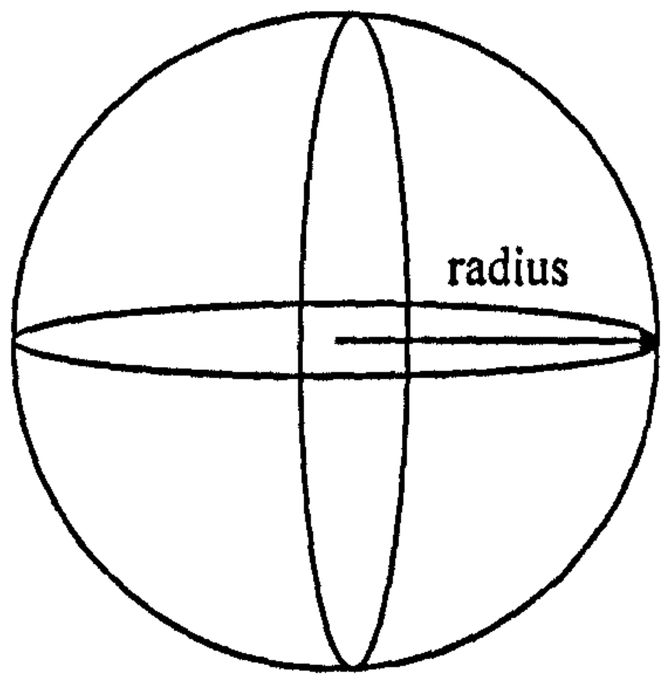
Cylinder {

bottom	TRUE
height	2
radius	1
side	TRUE
top	TRUE

}

Figure 4-8 The Cylinder node.

4. **Sphere :** The ***Sphere*** node specifies a sphere centred at (0, 0, 0) in the local coordinate system. The radius field specifies the radius of the sphere and shall be larger than 0.0. Figure 4-9 depicts the fields of the Sphere node.



Sphere { radius 1 }

radius = 1.0

Figure 4-9 The Sphere node.

5. **IndexedFaceSet:** The ***IndexedFaceSet*** node represents a 3D shape formed by constructing faces (polygons) from vertices listed in the coordfield. After being exported into VRML 2.0, most objects

modelled by 3D Max studio or AutoCAD are described by the IndexedFaceSet node.

6. **ElevationGrid:** The ***ElevationGrid*** node specifies a uniform rectangular grid of varying height in the Y=0 plane of the local coordinate system. The geometry is described by a scalar array of height values that specify the height of a surface above each point of the grid. The ElevationGrid node is a good candidate for the floor or terrain of the scene since it provides better compression than an IndexedFaceSet and thus shorter download time.

According to the capabilities of geometric modelling provided by VRML, it is obvious that separating out the properties into different nodes results in VRML files a little bigger although the prototyping mechanism can be used to reduce file size. However, for complex objects with curve surfaces and irregular shape, these capabilities are insufficient to handle them and impossible to create them just by using a simply text editor (Gueziec et. al. 1999). These are the constraints of geometric modelling in VRML.

4.2.4 Developing 3D models of virtual facilities in VR database by integrating with CAD/CAM packages - Deneb's Virtual NC and QUEST

In this research project, the major 3D modelling techniques and VRML geometric modelling capabilities described in above sections are combined for modelling 3D objects to create the VR database by the author in the proposed Web-based VR system.

In a VR system, a 3D geometrical model of a facility created according to a real one is incorporated into a virtual environment for applications in design and

manufacturing. A real facility (e.g. machine) consists of many components, which are made up of different parts. For example, a conventional milling machine comprises a base, a working table, a head, main shaft, a servo motor and a spindle motor, etc. Thus, a 3D machine model is constructed by a part-component-machine procedure, and then is imported into a virtual machine library.

A hybrid method that combines the features of VRML with some advantages of commercial modelling packages has been proposed to model 3D objects in this project by the author. Figure 4-10 illustrates the hybrid modelling method to develop 3D models of facilities in VR database.

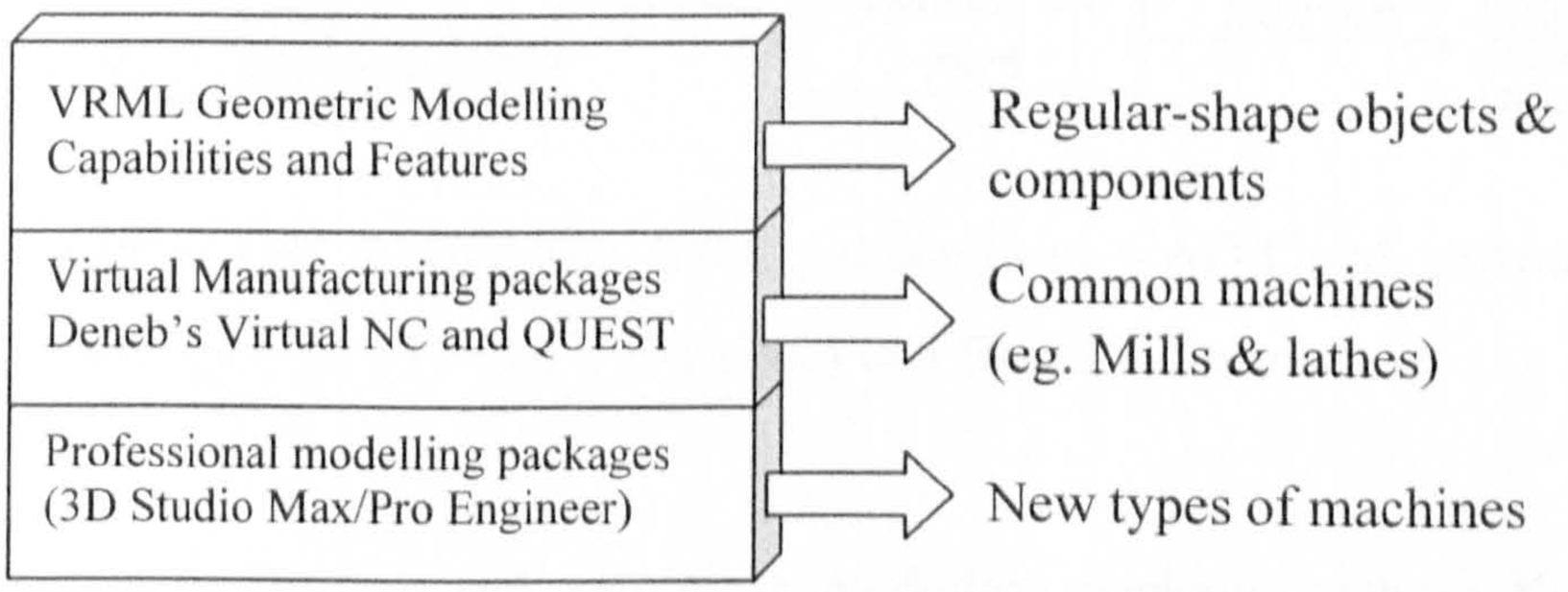


Figure 4-10 The hybrid modelling method to develop 3D models of facilities in VR database (proposed by the author)

Since geometric modelling in VRML 2.0 is mainly based on geometric primitives such as cubes, spheres, cylinders, cones, etc, it is basically used to model some components whose shapes are regular. Meanwhile, various modelling methods described in Section 4.2.1 have been integrated by the author to shorten the system development cycle. Some common machine models such as mills and lathes have been borrowed from a machine library provided by some professional CAD/CAM packages such as Deneb's Virtual NC and QUEST. QUEST and Virtual NC have been used for manufacturing applications for more than three years. They are high-end special effects tools and include the libraries of virtual

machines. With the outstanding capabilities of QUEST and Virtual NC, it is the choice for anyone wanting a specific virtual machines library.

In this project, in order to reuse the models provided by QUEST and Virtual NC as components of the virtual environment generated by VRML, the model files from QUEST and Virtual NC are saved as IGES format. Then, they are converted into VRML 2.0 world files by a translation tool (e.g. 3D Viewer). These files in VRML 2.0 can be incorporated with World Wide Web pages and be supported by any browsers on the Internet. Figure 4-11 shows the process of conversion.

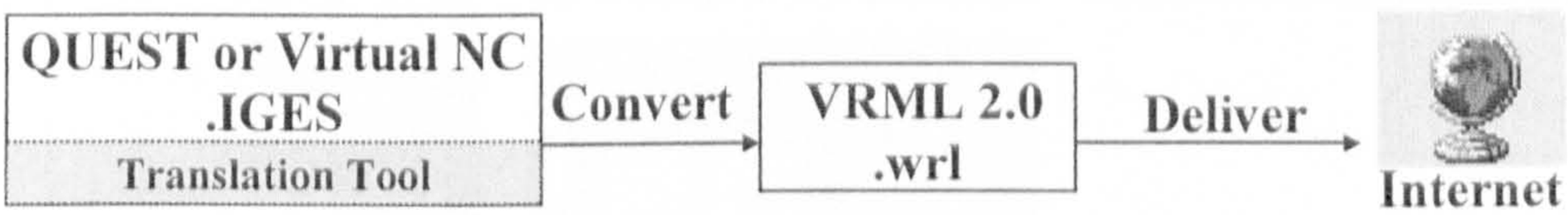


Figure 4-11 Converting IGES format from CAD/CAM software package (QUEST and Virtual NC) to VRML2.0 format

In addition, a professional geometric modelling package such as Kinetix' 3D Studio Max or Pro/Engineer is used to create some new types of machines. Similarly, the model files are converted into VRML 2.0 via a plug-in. However, the model's size could be very large because each polygon is converted into an *IndexedFaceSet* node description. In order to improve download and navigation speed, a model needs to be optimised by reducing the number of polygons required.

Figure 4-12 shows the 3D models of facilities created in VR database using the hybrid method. Due to the large size of geometric data of a 3D object, these 3D object files in VRML format are included in the CD attached to this thesis.

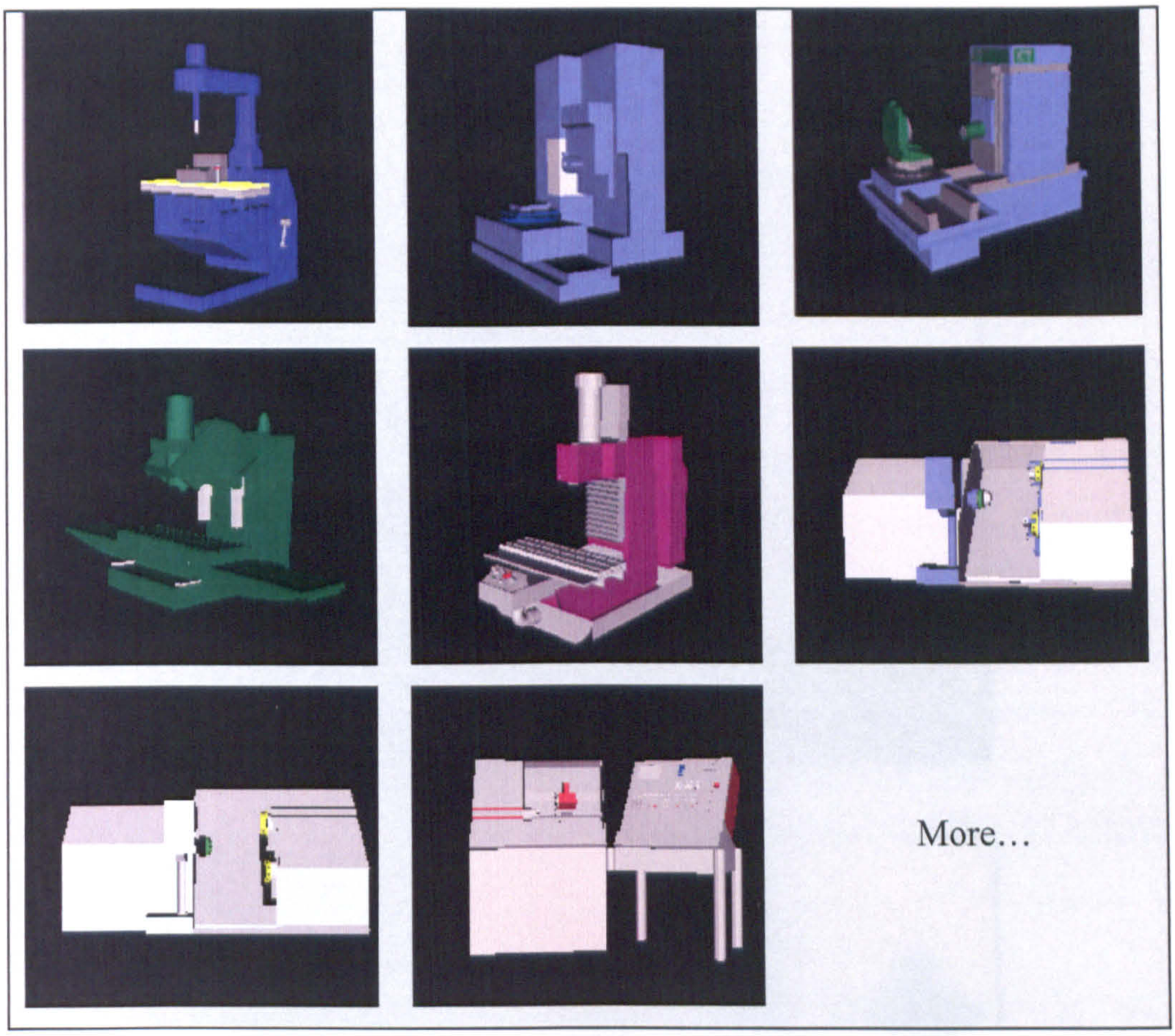
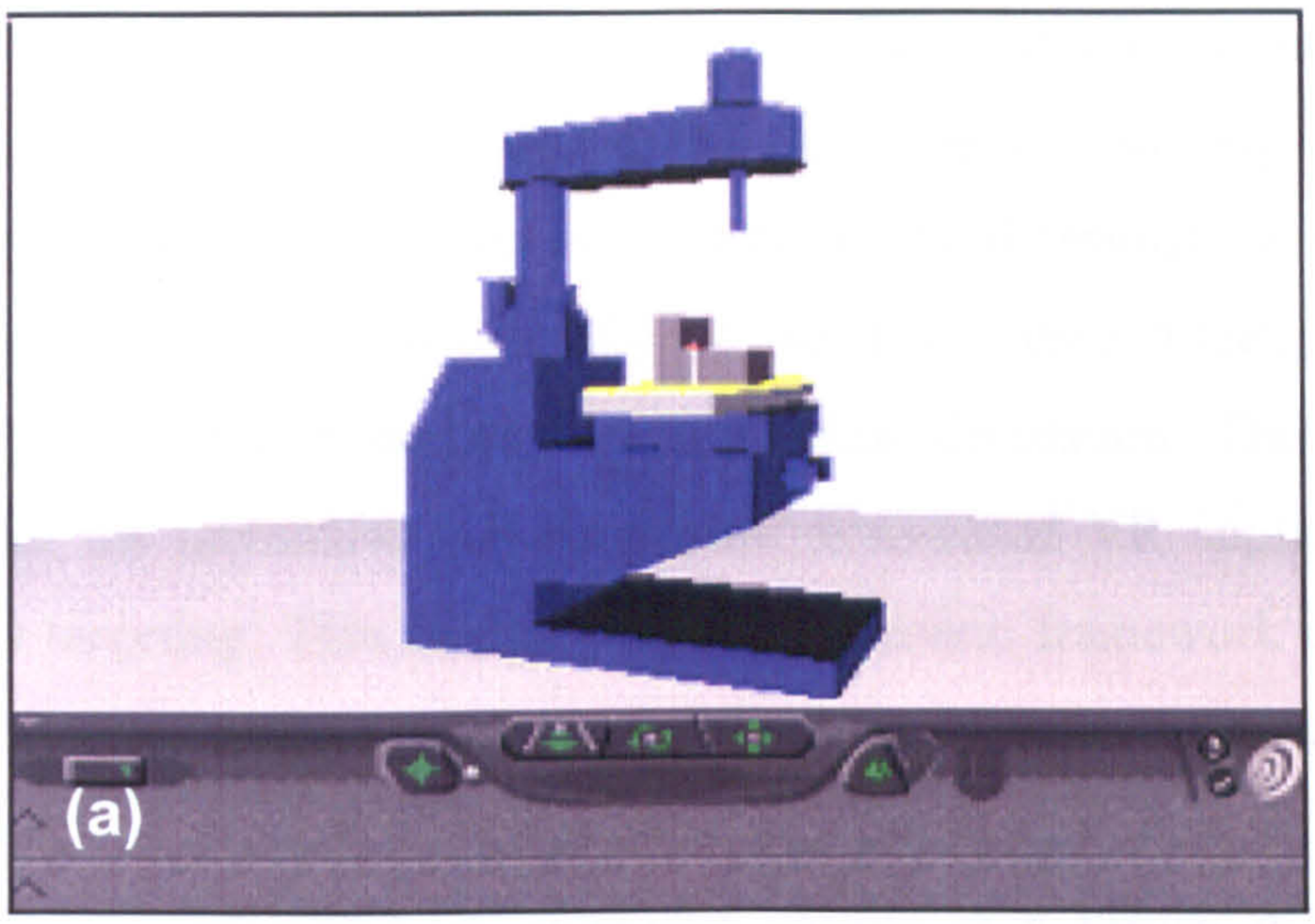


Figure 4-12 The 3D models of virtual facilities in VRML2.0 format in VR Database are created by integrating with professional CAD/CAM packages - Deneb's Virtual NC and QUEST

Since VRML files exist in a style of parent-children hierarchical structure. Models are subparts of the world. Such a structure makes it easy to create large worlds or complicated scenarios, which reuse subparts or inherit attributes and behaviours from subparts. As a result, not only a machine model but also its components or parts can be reused in various applications in design and manufacturing. For example, Figure 4-13a shows a milling machine's model is used in a machining application. Meanwhile, the model can be reused as a subpart in other applications such as workshop layout application shown in Figure 4-13b.



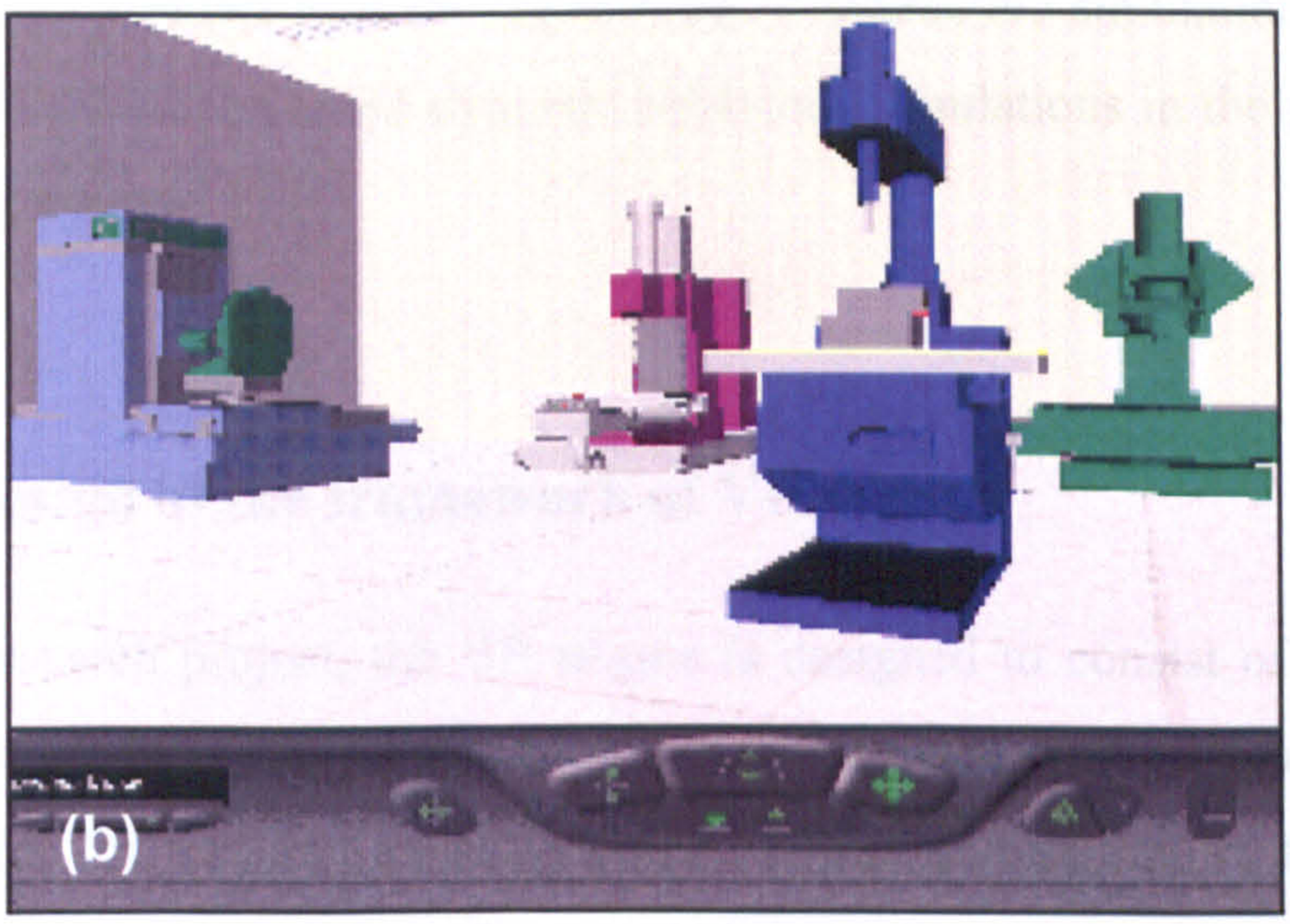


Figure 4-13 Reuse a 3D machine model in various applications in design and manufacturing (developed by the author)

(a) A mill model in the machining application.

(b) Reuse a mill model in the workshop layout application.

4.3 VR engine for interactive simulations

A VR engine is the cornerstone of the Web-based VR system in Net-VEs for interactive simulation. Most VR engines are based on state-of-the-art interactive real-time computer graphic algorithms and rapid rendering techniques for achieving the advanced performance, both in speed (frames-per-second) and in rendering quality. The performance of the VR engine directly influences its capability of real-time display and interactive simulation. This is particularly critical for the interactive and distributed Web-based VR applications that the project is targeting. This section proposes a generic framework of a VR engine and designs the model of interactive simulation for the proposed system. The functionality of simulation engine deals with the mechanisms for simulating and controlling the behaviour of geometric objects in a virtual environment. The section then presents how the capabilities of interactive simulation in VRML are used to develop interactive dynamic behaviour simulations in the proposed Web-based VR system.

4.3.1 Design of the framework of VR engine

In this research project, the VR engine is designed to consist of a 3D graphics engine, a simulation engine, and a processing system. The 3D graphics engine is responsible for mapping, creating texture, lighting, rendering, and displaying the realistic images in real time. The simulation engine has the ability to handle real-time interactive simulation such as object dynamics and physical constraints, collision detection, Level of Details (LOD) and so on. The processing system deals with a huge amount of data calculation associated with the graphic and simulation engines. Meanwhile, it is also responsible for coordinating various I/O controls supplied by users and communication among multiple users involved in the Net-VEs. The generic framework of the VR engine is designed by the author as shown in Figure 4-14.

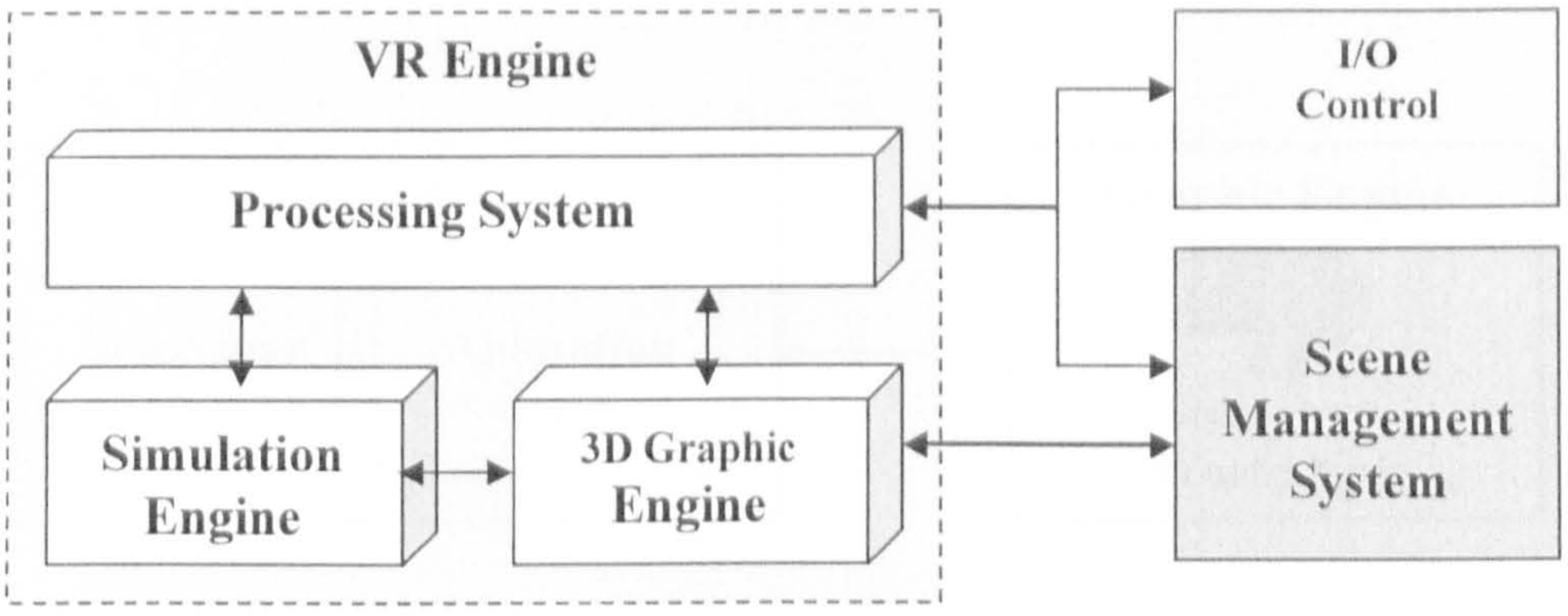


Figure 4-14 The generic framework of VR engines (designed by the author)

The engine can perform between 20-35 simulation steps per seconds for real-time display. Furthermore, the VR engine must be able to interface with the scene management system to directly manipulate the geometric objects. Normally, the VR engine can be implemented on the multiprocessor graphics workstations. With the advent of graphics accelerators, sufficient graphics capabilities have become available on standard PCs.

4.3.2 Design of the model of interactive simulation

The functionality of the simulation engine deals with the mechanisms for simulating and controlling the behaviour of geometric objects in the virtual world (Fishwick 2000). Each object is assigned a set of attributes. Some attributes control the visual appearance of the objects, while others control its behaviour. It also maintains the global timer/controller, sends commands to the visual interface to update the displayed scene. The graphic engine maintains abstract relationships between objects and translates behaviour into scene-graph changes. Using the virtual environment as a 3D visual interface to the interactive simulation, the link between geometry and simulation models should be bi-directional. The generic model of interactive simulation in VR engines is designed by the author as shown in Figure 4-15.

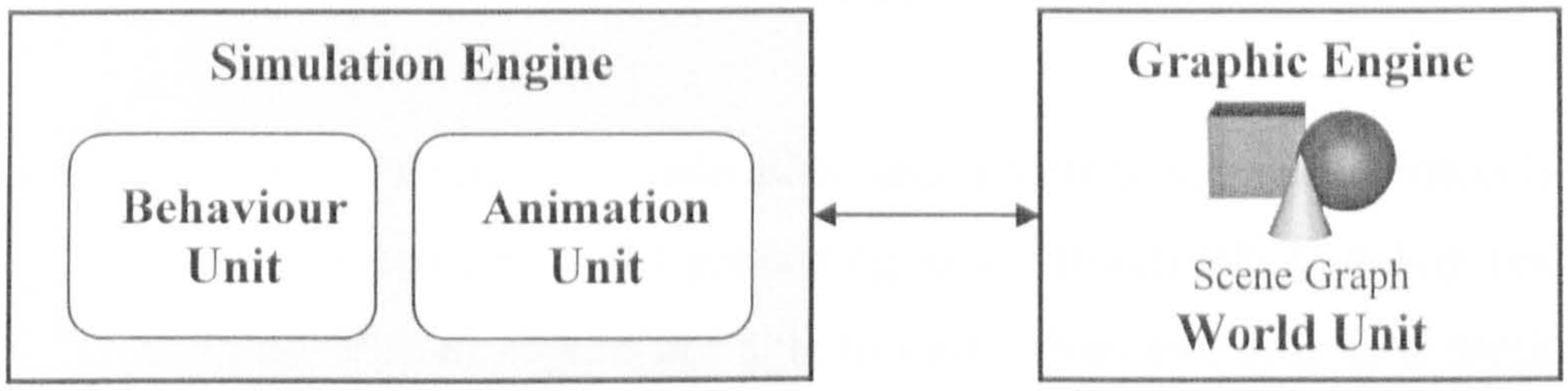


Figure 4-15 The generic model of interactive simulation (designed by the author)

The model is designed to consist of a ***World*** unit that is manipulated in real-time, an ***Animation*** unit which utilizes descriptions of primitive animated actions (such as the movements of conveyer) to manipulate the geometry in scene graphs, and a ***Behaviour*** unit which supports the computation of behaviours for objects and translates objects' behaviours into the world coordinates of the visual interface. For example, if a work-piece should be moved to another workstation, both start and finish positions of the animation must be calculated in the virtual environment. At run-time, virtual objects' movements and behaviour are computed by iterating an update cycle that alternates between the ***Animation*** and ***Behaviour*** Models. The key components are described in details as below:

- **World unit - Scene Graph:** In a virtual environment, a complex scene is made up of a number of objects which are based on the scene graph in a hierarchical tree as described in Section 4.2.2. The 3D objects are positioned in a scene by applying various transformations such as translation, scaling or rotation. The Scene Graph representation is the basis of the rendering cycle in the visual interface which produces the real-time display. However it is not a good representation for supporting cause and effect relationships or object identity and attributes. If a work-piece is to be moved onto a workstation, the polygons associated with these objects have to undergo the correct changes in the scene graph. For

this to happen, the mapping between the conceptual categories of work-piece and workstation has to be mapped onto the correct parts of the Scene Graph. This mapping is maintained in 3D Graphic Engine.

- **Animation unit:** The ***Animation*** unit controls the virtual objects and provides mechanisms for generating and interactively blending realistic motions. Virtual objects are able to move from one animated motion to another in a smooth and natural fashion in real time. Motions can be layered and blended to simulate the advanced dynamic behaviours.

- **Behaviour unit:** The ***Behaviour*** unit is responsible for ensuring that the behaviour of objects is correctly modelled, maintained, and mapped between the scene graph and hence translating the interactive simulation into a form that can be handled in the scene graph.

4.3.3 Examining interactive simulation in VRML

In order to apply the above model of interactive simulation in the proposed Web-based VR system, the section discusses interactive simulation techniques in VRML to develop interactive applications in design and manufacturing.

To achieve interactive simulation in virtual environments, VRML allows using "***Sensors***" to create events that are passed to program scripts for implementing dynamic behaviours, modifying a material, modifying geometry, etc. Most of VRML sensors generate events as the result of a user action (e.g. clicking on an object) or user proximity. In addition to these, the ***Collision*** group node and the ***Anchor*** node can also be used to detect user interaction and generate events. In this project, advanced interactive simulations are created using a combination of TouchSensors (to generate initial events), Scripts (to determine what to do with them) and TimeSensors to play resulting animations. For example, to create an interactive door, a TouchSensor could be attached to a door object and the

touchTime be routed to a Script so that the Script generates an event to start an animation to either open or close the door depending on whether the door was open or closed when it was clicked on.

4.3.3.1 Sensors for built-in functionality of simulation

In order to simulate dynamic behaviours in a virtual environment, it is essential for the VR engine to provide the mechanism of handling events passing. In VRML, events are in the style of data flow from the eventOut field of one node to an eventIn field of the same datatype of another node. The receiving node may respond to the event by processing it, updating it, and generating further events. The standard VRML97 sensors are described as follows, including: *TouchSensor, TimeSensor, CylinderSensor, PlaneSensor, SphereSensor, ProximitySensor, and VisibilitySensor.*

- **TouchSensor**

  TouchSensor detects events from a pointing device such as mouse. It has six eventOut fields that generate events depending on what user actions are sensed. The most commonly used of these are:

  a) isActive - generates TRUE (SFBool) when the mouse button is pressed on a sensed object and FALSE when it is released.
  b) isOver - generates TRUE (SFBool) when the cursor is over a sensed object and FALSE when the user moves the cursor off the sensed object.
  c) touchTime - when the user releases the mouse button (ie when isActive generates a FALSE event), an SFTime event is generated containing the absolute time.

  In addition to the eventOut fields above, the TouchSensor has an 'exposedField' called 'enabled' which can be set to TRUE or FALSE. This can be used to switch the sensor on/off by routing SFBool events to it. The default value is TRUE so to create an active TouchSensor.

- **TimeSensor**

  The TimeSensor is a timer such as a clock that generates events. The TimeSensor's eventOut fields are used to route events from the TimeSensor to other nodes.

  a) isActive - generates a TRUE event when the sensor becomes active and starts to output events, and FALSE when it stops.
  b) cycleTime - generates the absolute time value each time a cycle starts over.
  c) time - generates an absolute time value continuously, as long as the sensor is generating fraction_changed events
  d) fraction_changed - generates a SFFloat value between 0.0 and 1.0 as the clock progresses through the cycle.

- **CylinderSensor, PlaneSensor, and SphereSensor**

  These sensors are used to detect mouse drag motion that can be used to translate objects depending on user actions.

  a) The **CylinderSensor** generates SFRotation values around the Y axis, as if the user is rotating a cylinder.
  b) The **SphereSensor** generates SFRotation values, as if the user is rotating a ball.
  c) The **PlaneSensor** outputs values in the XY plane of the sensor's parent coordinate system when the sensor is active.

  The values output from these sensors can be routed directly to a Transform group, enabling the user to drag the sensed object around (PlaneSensor), rotate it freely (SphereSensor) or rotate it around the Y axis (CylinderSensor).

- **ProximitySensor, VisibilitySensor, and Collision**

  These sensor nodes generate events based on the user's location in the virtual environment as opposed to mouse clicks or mouse drag motion.

a) The **ProximitySensor** generates events if the user is within a user defined distance from a sensed group. Its eventOut fields include isActive, enterTime, exitTime, position_changed, and orientation_changed. The position and orientation changed values are positions within the sensed region, in the coordinate system of the ProximitySensor.

b) The **VisiblitySensor** senses whether a boxed shape area is visible to the user. EventOut fields are isActive, enterTime, and exitTime.

c) The **Collision** node is a group node that generates a collideTime event when the user's avatar collides with the scene's geometry.

4.3.3.2 Defining constraints

By defining constraints to the transformation of an object, the developer can control how the object can be interacted with in simulations. As the above description, **PlaneSensor, SphereSensor** and **CylinderSensor** can be used to directly manipulate objects. The developer can constrain the effect of these sensors by setting exposedFields for these sensors.

- The **PlaneSensor** has *minPosition* and *maxPosition* that are used to specify the X and Y translation limits:

```
PlaneSensor {
minPosition -1 -1
maxPosition  1  1  }
```

- The **CylinderSensor** has *maxAngle* and *minAngle* that limit how far you can rotate:

```
CylinderSensor {
maxAngle 1.571
minAngle 0.0     }
```

- The **SphereSensor** has no built-in constraints. Thus, the developer needs to use Scripts to constrain a SphereSensor. Arbitrarily complex constraints can be implemented using Script nodes. For example, a SphereSensor's rotation_changed value can be routed to a Transform group via a Script node instead of directly, and the Script can then determine what values are permitted to be passed on to the Transform group. Scripts can also be used to determine when to enable/disable sensors, adding further constraints to the user's interaction with the virtual environment. For example, clicking on a button could cause a script to enable a PlaneSensor, allowing another object to be moved around, while clicking on the button again could lock the object in position by disabling the PlaneSensor.

4.3.3.3 Integrating scripts for advanced simulation

Scripts allow creating advanced animations on basis of mathematical expressions. They are used to calculate motion paths, access databases, integrate Artificial Intelligence (AI) objects and so on. A VRML **Script** node consists of an interface and a program script. The program scripts in a Script node are like small applications, written in JavaScript or Java (Ghee et. al. 1995).

- **The interface**

  Field data has a type, name and initial value. The developer can access and modify the field values from the program script. An **eventIn** has a type and name. For each eventIn, the developer needs to create a function with the same name in the program script. The function is called when an event is passed to the Script node's corresponding eventIn field. An eventIn definition has the following syntax:

  eventIn *eventInType eventInName*

An **eventOut** has a type and a name and enables the Script to generate events that can be passed to other nodes. An eventOut definition has the following syntax:

eventOut *eventOutType eventOutName*

- **The structure of program script**

Having defined an interface, the dynamic behaviours associated with simulations can be added by a program script. The program script is provided as the value of the Script node's 'url' field, and can either come directly in the url field's value, if the value starts with "javascript:", or it can be defined in another file (e.g. "script.js"). In addition, some large scripts can be stored in an individual file. The Script node can be summarized as follows:

```
DEF ScriptName Script {
  field fieldType fieldName initialValue
  eventIn eventInType eventInName
  eventOut eventOutType eventOutName
  url "javascript:
    function eventInName(value, timestamp) {
      ...
      eventOutName = ... ;
    }"
}
```

- **Programming scripts in JavaScript and Java**

JavaScript is convenient to use for scripting VRML worlds. Java can be used to implement more advanced scripts when JavaScript is inadequate, e.g. scripts that access databases or need to perform low-level networking tasks.

4.3.4 Developing interactive dynamic behaviour simulation in the Web-based VR system

A dynamic machine and a process flow need to be simulated according to the level of detail required. The VR engine plays an important role to generate the animations. In this project, the following mechanisms of VRML (Carey and Bell 1997) have been used to develop interactive dynamic behaviour simulation.

- **Interpolator nodes**

  Perform keyframe animations. According to a list of specified keyframe values and times, the VRML browser will automatically interpolate the "in-betweens". There are several different types of interpolators: ***PositionInterpolator***, ***OrientationInterpolator***, ***CoordinateInterpolator***, etc. Each one animates a different field type. For example, the ***PositionInterpolator*** node is used to animate a cutting tool's position along a motion path. Then both ***PositionInterpolator*** node and ***OrientationInterpolator*** node have been employed to produce typical animated object motion. The first moves the object along a motion path, while the latter rotates the objects as it moves.

- **Sensor nodes**

  Detect and sense changes to the state of an input device, changes in time, or changes related to the motion of the viewer or objects in a virtual world. They include ***TouchSensor***, ***CylinderSensor***, ***SphereSensor***, ***TimeSensor***, ***VisibilitySensor***, and so forth. The ***TimeSensor*** node generates events as time passes and is the basis for all animated behaviours.

- **Script nodes**

  Allow the developer to define arbitrary behaviours by binding the Java and JavaScript languages. Although interpolator nodes do not support spline curve interpolation, a script can be programmed to directly implement the mathematics of the curve interpolation.

Generally, **Interpolator** nodes are essentially built-in scripts that perform simple animation calculations. They are usually combined with a ***TimeSensor*** and some node in the scene graph to make objects move. For example, the moving of the working bed of a mill along *x*-axis in specified time can be defined by using ***PositionInterpolator*** node and ***TimeSensor*** node. Figure 4-16 shows an example developed by the author using this method.

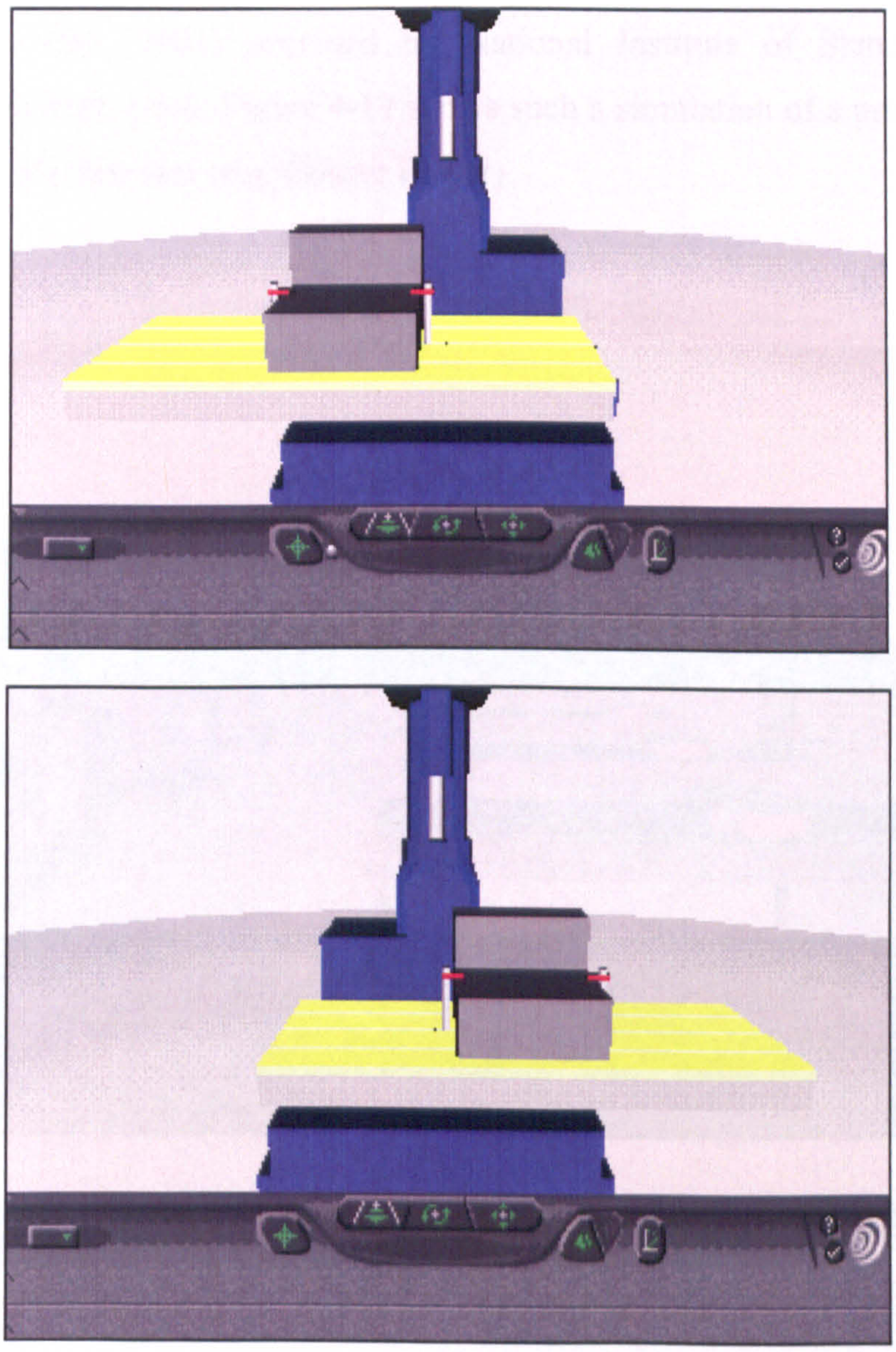


Figure 4-16 The simulation of the working bed of a mill moving along *x*-axis (developed by the author).

In addition, Physical Based Modelling (PBM) is an important method to simulate animations for virtual environment. PBM can be used in simulation engine to generate highly actual animations (Baraff and Witkin 1997).

Since some professional manufacturing simulation packages such as Deneb's QUEST can directly emulate real-world system behaviours, its simulation capabilities (QUEST User Manuals 1995) could be coupled to VRML worlds via a translator (NIST 2001) provided by National Institute of Standards and Technology (NIST), USA. Figure 4-17 shows such a simulation of a process flow through a VRML browser (e.g. Cosmo Player).

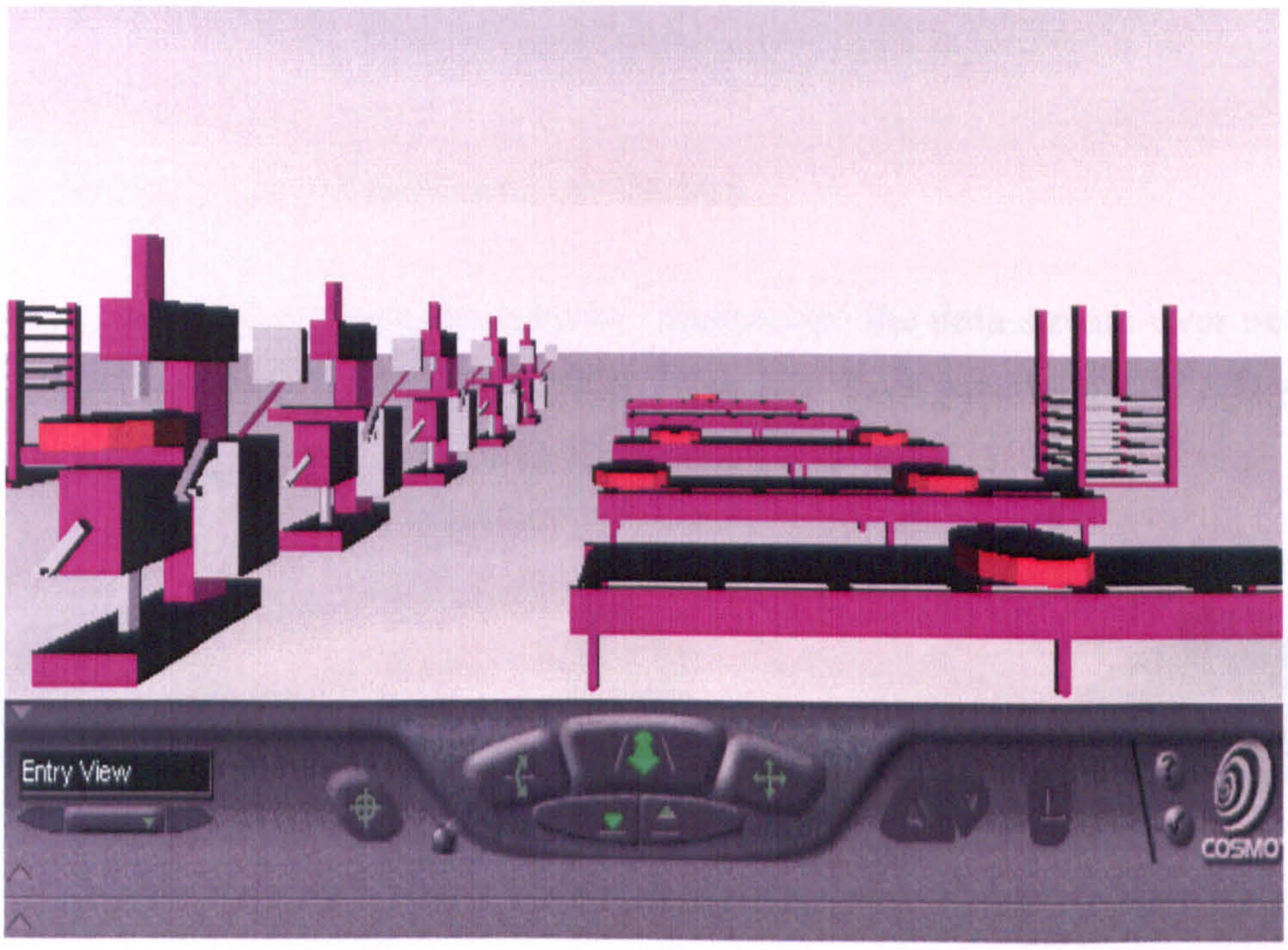


Figure 4-17 A view of the simulation of a process flow through the VRML browser (e.g. Cosmo Player)

4.4 Network Communications

Since Web-based VR applications will involve multiple users connected through the Internet in a distributed environment, they require considerable network bandwidth and performance for communication. This section firstly describes the networking fundamental issues involved in the design of Web-based VR systems, including network latency; network bandwidth, network reliability, and network protocol. Then, it examines the Internet protocols — IP unicasting and IP multicasting based on different network architectures. In particular, this section evaluates network protocols by comparison of their advantages and limitations in order to select the most appropriate protocol for different types of Net-VE architecture design. Finally it suggests a hybrid method of reliable communication by using multiple protocols for the proposed system in Net-VEs.

4.4.1 Networking fundamental issues

There are some basic concepts used to characterize the data stream over network, as shown in Figure 4-18, including: *Network Latency*, *Network Bandwidth*, *Network Reliability* and *Network Protocol* (Singhal and Zyda 1999).

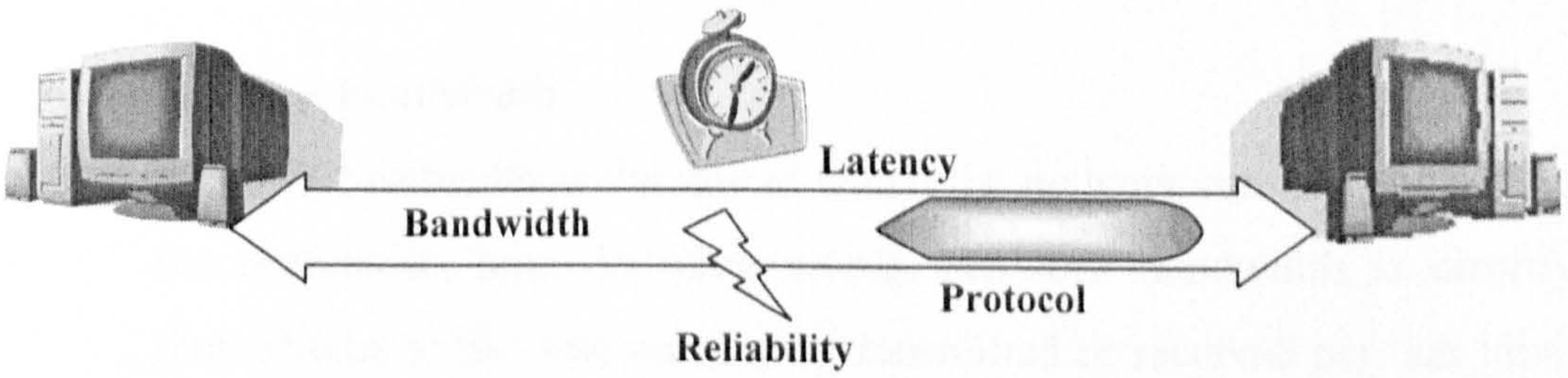


Figure 4-18 Network communications are characterized by bandwidth, latency, reliability, and protocol.

- **Network latency**

  Network latency, a synonym for network delay, is an expression of how much time it takes for a data packet to get from one designated point to

another. Such network delay is one of the biggest challenges in the design of Net-VEs applications since delay directly impacts the sense of presence in Net-VEs. The contributors to network latency include:

a) **Propagation:** This is simply the time it takes for a data packet to travel between one place and another at the speed of light.

b) **Transmission:** The medium itself such as optical fibre and wireless introduces some delay. The size of the data packet introduces delay in a round trip since a larger packet will take longer to receive and return than a short one.

c) **Router and other processing:** Each gateway node takes time to examine and possibly change the header in a data packet.

d) **Endpoint computers:** Within networks at each end of the journey, a data packet may be subject to storage and hard disk access delays at intermediate devices such as switches and bridges. Latencies within modems and network hardware represent some of the most significant sources of latency in Net-VEs.

- **Network bandwidth**

  Network bandwidth is the rate at which the network can transfer data to the destination host. In other words, Network bandwidth is directly proportional to the amount of data transmitted or received per unit time. Network bandwidth is expressed as data speed in bits per second (bps). The network bandwidth is determined by the type of connections used to deliver data, and it is also limited by the hardware used to transmit the data (Stallings 1996). Thus, network bandwidth often represents the rating of modem or broadband Internet service. For example, a modem can typically handle data at the rate from 14.4Kbps to 56Kbps. The latest

Ethernet can transfer 100Mbps. The highest bandwidth fibre optical cables can handle data up to 10Gbps.

However, network bandwidth can refer to both actual and theoretical throughput, and it is important to distinguish between the two. For example, a V.90 modem supports 56 Kbps of peak bandwidth, but due to limitations of the telephone lines and other factors, it is impossible for a home dial-up network to actually achieve this level. Likewise a Fast Ethernet network theoretically supports 100 Mbps of bandwidth but this level can never be achieved in practical use thanks to overhead in the hardware and in the computer's operating system.

Essentially, network bandwidth represents the capacity of the network connection, and it is obvious that the greater the capacity, the more likely that greater performance will follow. In a qualitative sense, bandwidth is proportional to the complexity of the data for a given level of system performance. For example, it takes more bandwidth to download a photograph in one second than it takes to download a page of text in one second. Large sound files, computer programs, and animated videos require still more bandwidth for acceptable system performance. Virtual reality (VR) and full-length three-dimensional audio/visual presentations require the most bandwidth of all.

- **Network reliability**

  Network reliability is the ability of a network to maintain or restore an acceptable level of performance during network failures by applying various restoration techniques, and mitigation or prevention of service outages from network failures by applying preventive techniques. On the other hand, network reliability is a measure of how much data is lost by the network during the transmission from source to destination host. This data loss can be divided into two categories as below.

a) **Data dropping:** the data does not arrive at the destination host at all because it has been discarded by the network.

b) **Data corruption:** the content of the data packets has been changed during transmission so that the arriving data packet is basically useless to the destination host.

There are various causes of data loss that influence network reliability, including:

a) **Transmission:** Data might be corrupted when the network itself is poorly connection or are subject to electrical or magnetic interference. The data loss is most severe in wireless connection situations, either between wireless modems or between the ground and a satellite due to competing transmissions, bad weather etc.

b) **Network routers:** The most common causes of data loss are the network routers that deliver data between transmission lines. Routers can process packets at a certain rate, but data does not tend to arrive at a steady rate. Sometimes, the coming data packets are so many so that the router cannot handle them due to its limited data memory. The router has to discard some of the queued data. This results in the data dropping. Such data dropping exists widely. For example, at peak times during the day, packet loss caused by dropping can be up to 50%.

- **Network protocol**

  Network protocol defines a common set of rules that two applications on the network use to communicate with each other. According to (Singhal and Zyda 1999), a protocol consists of the following components:

a) **Packet formats:** Since the two endpoints of a communication connection require understanding the packets from each other, the packet formats describe what each type of packet looks like. Sometimes, the data compression methods may be involved.

b) **Packet semantics:** The packet semantics of a protocol are typically described using a Finite State Machine (FSM) which represents the transition involved in the protocol. It is useful to determine how the sending application will indicate that it has finished sending a message, and how the receiving device will indicate that it has received a message.

c) **Error checker:** If a host receives a packet that is improperly formatted, the error checker is able to determine how deal with the error packets according to the type of error checking being used.

There are a variety of network protocols in use. One of the most popular protocols for LANs is called *Ethernet*. Another popular LAN protocol for PCs is the *IBM token-ring network*. Each has particular advantages and disadvantages; for example, some are simpler than others, some are more reliable, and some are faster. Some protocols are particularly designed for different applications.

From a user's point of view, the only interesting aspect about protocols is that the user's computers or applications must support the right ones if the user wants to communicate with other computers or applications. The protocol can be implemented either in hardware or software.

4.4.2 Examining Internet protocols

The Internet is often described as being based on TCP/IP. At a fundamental level, the Internet Protocol (IP) is relatively unreliable because it does not guarantee the

delivery of any particular packet of data. The Transmission Control Protocol (TCP) which runs above the IP provides mechanisms to compensate for this potential loss of packets by adding careful error correction by marking data with sequence numbers (Stevens 1994). Thus, TCP/IP forms a reliable connectivity solution for the Internet to support most traditional Web-based applications such as E-Mail, Web browsing (HTTP), File Transfer Protocol (FTP), remote login and so forth.

4.4.2.1 Unicasting

Unicasting is where a separate connection is set up between a server and each of its clients. TCP is a unicasting protocol which means it is strictly one-to-one communication. This type of connectivity is optimal for situations where every client has specific needs such as browsing Web pages and reading particular E-Mail messages.

- **Client–Server architectures**

  Client-server architectures are sometimes called *two-tier architectures.* It is a network architecture in which each computer on the network is either a *client* or a *server*. Servers are powerful computers dedicated to managing disk drives (*file servers*), printers (*print servers*), or network traffic (*network servers*). Clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

  Figure 4-19 shows a unicasting system based on client-servers architecture. The system involves N clients connected to a standard Web or FTP server. Each client requires P packets per second. So the server requires that network bandwidth is

$$\text{Network Bandwidth} = N \times P$$

Bandwidth needs increase linearly with the number of clients (Fisher 2002).

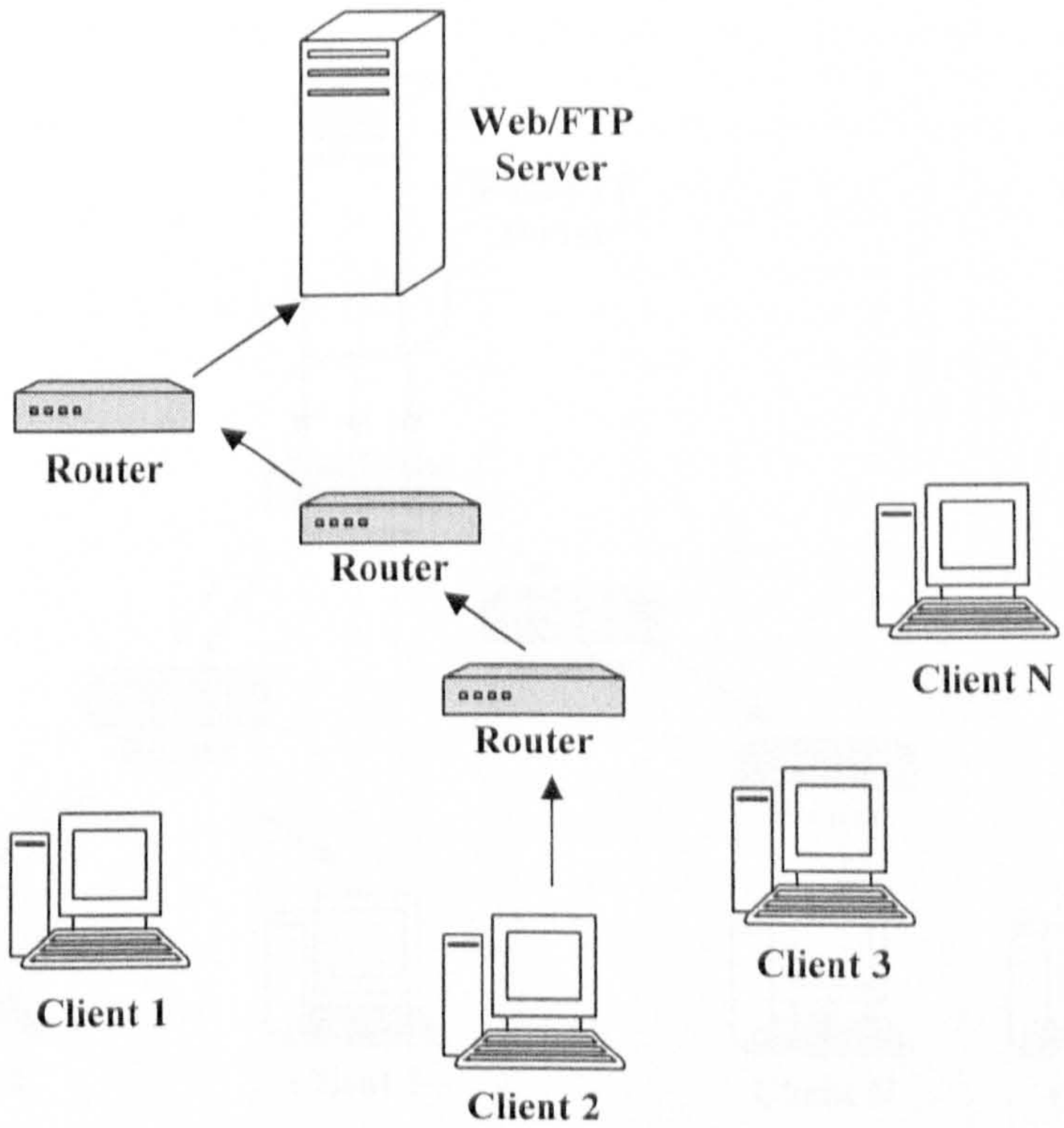


Figure 4-19 Server-based unicasting system. Web/FTP sever sends packet to client hosts.

A collaborative virtual environment which is built on top of the client-server architecture is shown in Figure 4-20. Each client generates changes to the shared environment at rate P. The server must then transmit the changes to every other client in the CVE. Thus, each client transmits at P and receives

$$(N-1) \times P$$

The server, however, receives

$$N \times P$$

and transmits

$$N \times (N-1) \times P$$

The network bandwidth required by the server grows exponentially with the number of clients (Fisher 2002).

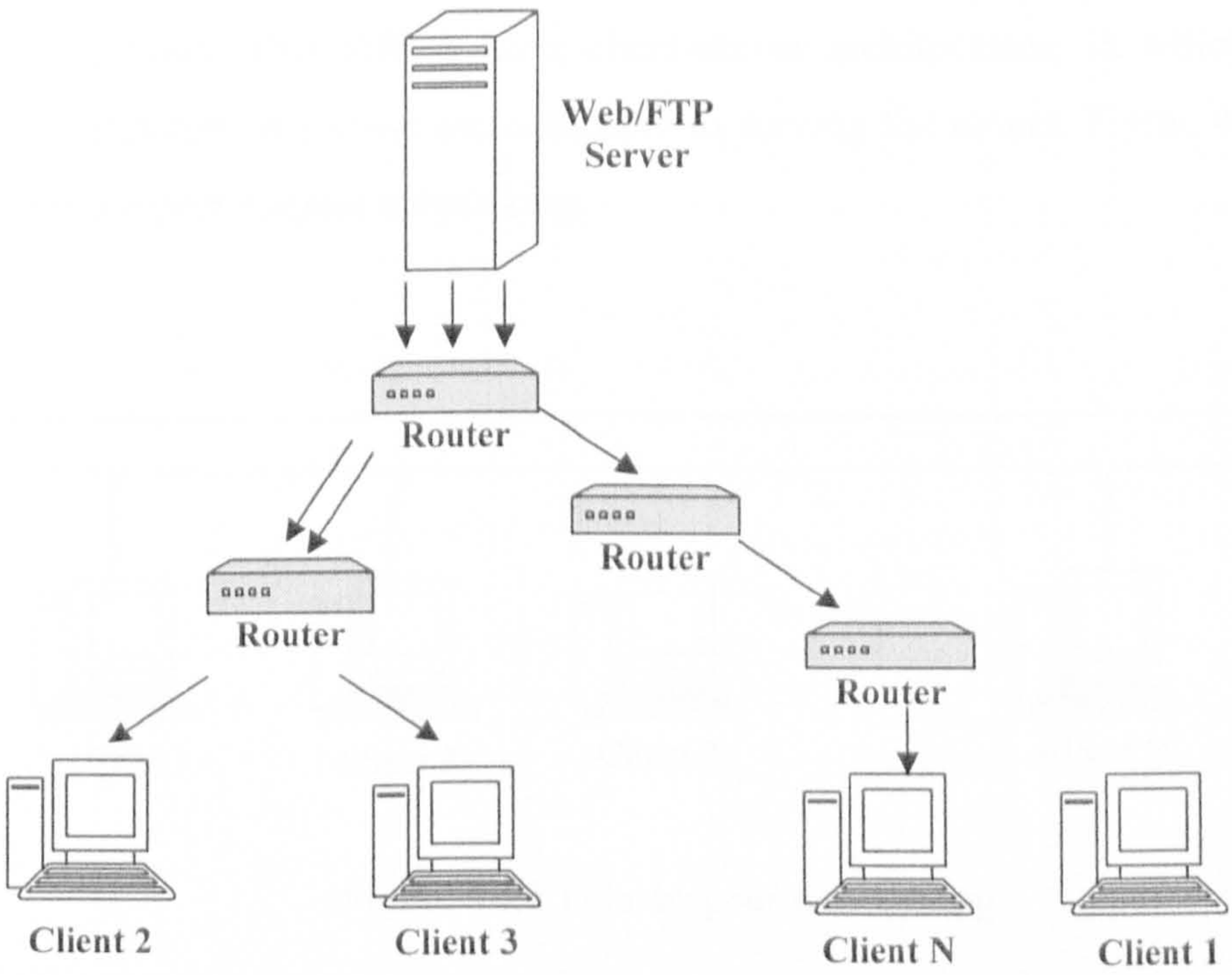


Figure 4-20 Server-based CVE transmit changes to the shared environment to the server. The server receives the changes, then sends them to all members.

Any system that requires the server to inform clients of what other participants are doing such as online chat systems and stock trading results in the same exponential bandwidth growth. Net-VEs suffer more than most because they have to handle large volumes of graphical data and meet real-time and interactive requirements. On the other hand, it is obvious that sending $N - 1$ identical copies of each update to all CVE participants is a waste of both server and network capacity (Fisher 2002).

- **Peer-to-peer architectures**

  Peer-to-peer is a type of network architecture in which each *node* (involved computers or workstations) has equivalent capabilities and responsibilities. This differs from client-server architectures, in which some computers as servers are dedicated to serving the others. Figure 4-21 shows a peer-to-peer networking.

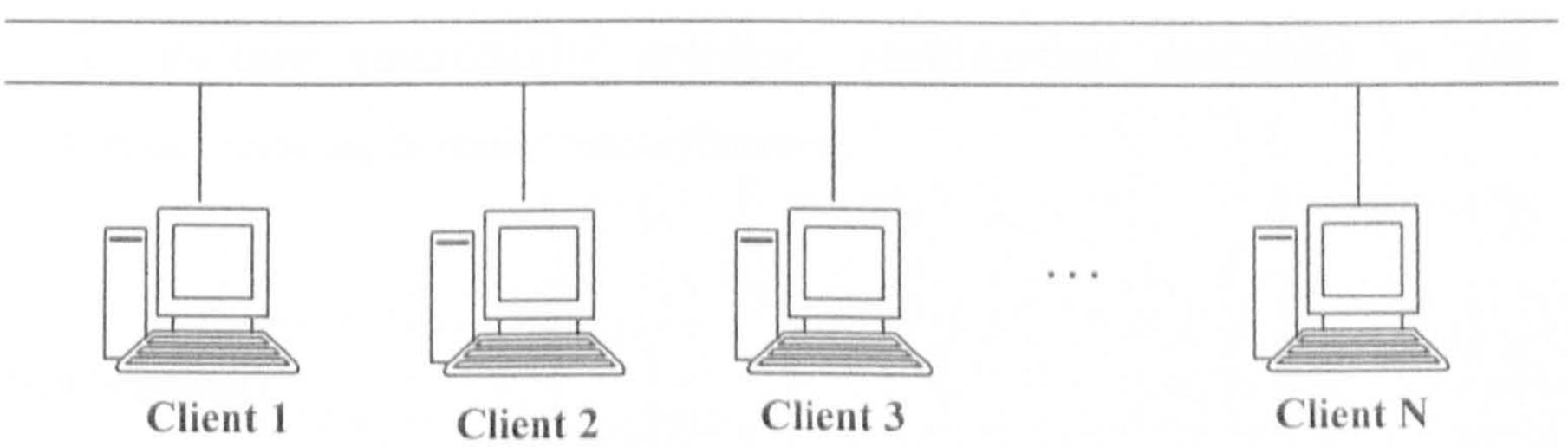


Figure 4-21 Peer-to-peer networking

The Peer-to-Peer architecture as an alternative networking approach has been adopted for CVEs (Singhal and Zyda 1999) in which every participant communicates directly with another member. In such peer-to-peer CVEs, all hosts must transmit at rate $(N-1) \times P$ and receive at $(N-1) \times P$, for a total bandwidth at each host is

$$2 \times N \times P$$

The Peer-to-Peer architecture avoids of a single massive bottleneck problem at the server existing in Client-Server networking through doubling the bandwidth at each host.

However, as N grows, it is difficult for hosts to enumerate all the hosts they must connect to. Moreover, many computers can realistically handle only a limited number of connections. In order to address this

shortcoming, the current existing CVEs (e.g. Massive-2) use *Area-of-Interest* techniques to divide participants into smaller spatial areas such as virtual zones (Greenhalgh 1996). Participants send updates only to those in the same area of interest, rather than to every participant. The *Area-of-Interest* techniques efficiently reduce the *N* in such peer-to-peer CVEs.

Peer-to-peer architecture is generally simpler, but it cannot offer the same performance under heavy loads. It is still inefficient for collaborative situations, where multiple hosts all want the same data at the same time. Thus, another connectivity solution, multicasting described in the following section, is more cost-effective.

4.4.2.2 Multicasting

Multicasting provides mechanisms that allow a server to transmit only a single data stream, regardless of how many clients might request it. Multicasting differs from broadcasting, which in IP networking means sending packets to every client whether they want them or not. Multicasting is being promoted by IP Multicast Initiative (IPMI) — a telecommunications industry consortium founded in 1996. This consortium consists of over one hundred members such as 3Com, Cisco Systems, Lucent Technologies, GTE Internetworking, and Stardust Technologies so forth (IPMI 2002). Multicasting has ability to provide one-to-many, several-to-many, and many-to-many network delivery services. It typically involves a high level of interaction as users communicate with each other as well as with the central server. A one-to-many example might be a live transmission of a technical conference. A several-to-many might be a panel interview between different locations. A many-to-many example might be a distributed research community with lots of members. Multicasting is useful for a variety of videoconferencing, audio and multi-user applications where numerous hosts need to communicate simultaneously. Thus, Multicasting provides a new communication patterns in Web-based VR applications.

With the introduction of Mbone embedded within the traditional unicast-only Internet, IP Multicast is an efficient networking model for large-scale Net-VEs which require combining multimedia issues such audio, video, 3D objects and so on (McCanne 1999). This section explores multicasting in the *Client-Server architecture* and the *Peer-to-Peer architecture* respectively.

- **The model of IP multicasting**

  IP Multicast extends the traditional unicasting delivery model of the Internet Protocol with efficient multipoint packet transmission, (Deering and Cheiton 1990) (Deering 1991). Using IP Multicast, a single data stream is sent to an arbitrary number of clients by copying it within the network at fan-out points along a distribution tree. Figure 4-22 show the model of IP Multicast using a distribute tree.

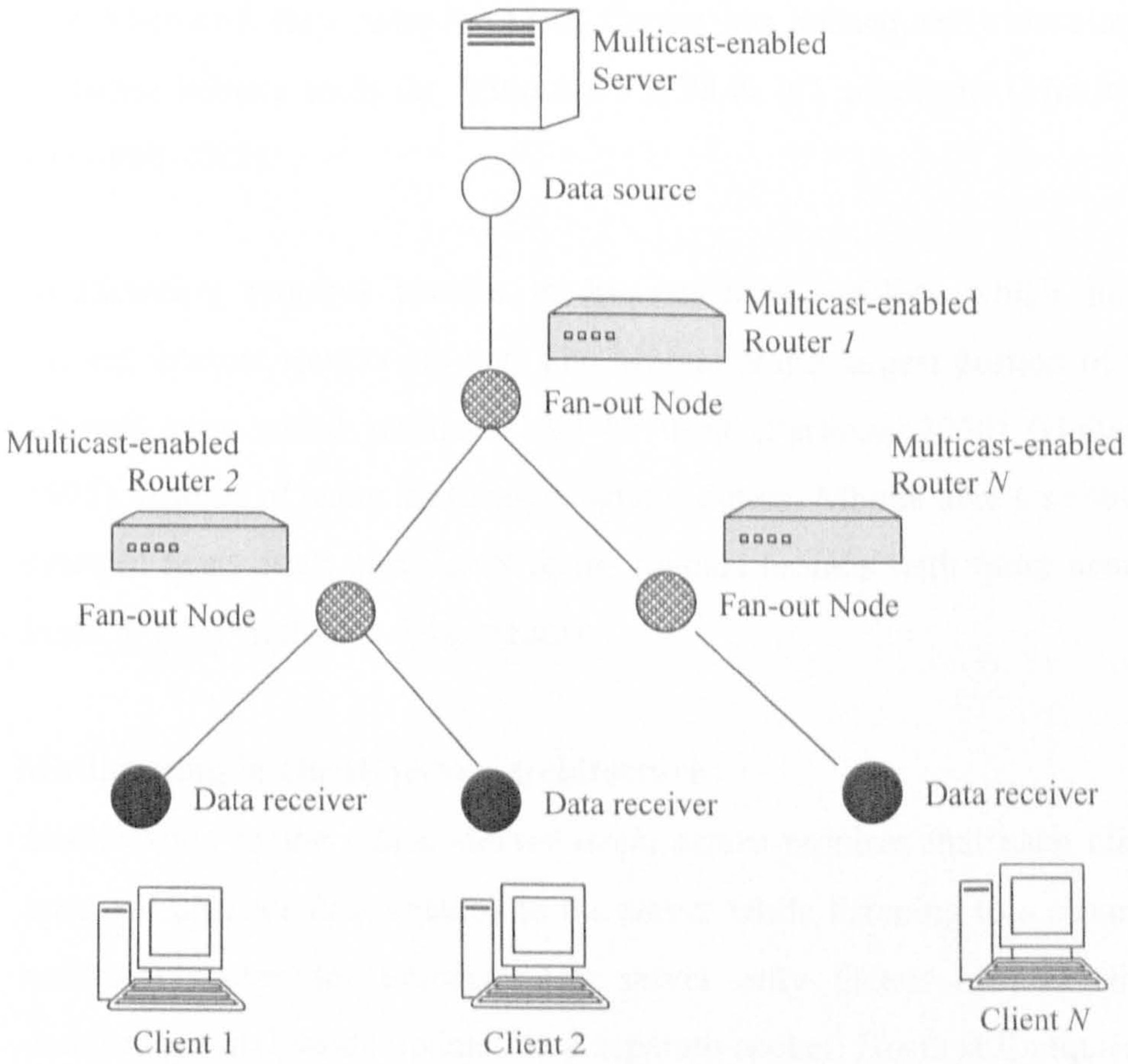


Figure 4-22 The model of IP Multicast using a distribute tree

Since copies of the data stream are only made to branches of the networking tree where clients that requested the data stream are located, this results in a dramatic reduction in overall bandwidth consumption. Thus, Multicasting is ideal for situations where clients want the identical data simultaneously. For example, the multicast backbone (MBone) is such a connectivity application on the Internet. It uses a network of routers — mrouters that can support multicast. These mrouters are either upgraded commercial routers or dedicated workstations with modified operating system kernels running in parallel with standard routers. It makes multicasting feasible on a global scale by taking advantage of installation of high bandwidth Internet backbone connections and widespread availability of workstations with adequate processing power (Rhyne 2000). The MBone software tool was firstly developed for UNIX workstations in Lawrence Berkeley National Laboratory (LBNL 2002). The Microsoft Bay Area Research Centre has subsequently developed freeware MBone tools for Windows 95, 98 & NT platforms (Microsoft Research 2002).

Multicasting requires routers to be multicast-enabled, which many current Internet routers are not. The Mbone is the largest portion of the Internet over which multicast can be used (Perlman 2000) (Huitema 1995). Instead of using multicast-enabled routers, Mbone uses a series of selected hosts from each LAN forms unicast tunnels with other nearby hosts to relay multicast data streams.

- **Multicasting in client-Server architecture**

Multicasting in the *Client–Server architecture* requires that each client host still unicasts data streams to the server while listening to a separate multicast socket for updates. The server only listens on the client connections and sends updates to a separate socket. Hosts still require P outgoing and $N \times P$ incoming, but not $N - 1$ (hosts also receive the updates they sent). The server receives $N \times P$, but now only has to

transmit $N \times P$ (rather than $N \times (N-1) \times P$), so bandwidth scales linearly rather than exponentially (Fisher 2002).

- **Multicasting in peer-to-peer architecture**

  In the peer-to-peer architecture, hosts multicast updates to a single group address. Each host needs only a single socket for transmitting and receiving, instead of $N-1$ individual unicast connections. Hosts require only P outgoing and $N \times P$ incoming as in the central server design and half the bandwidth required in the unicast architecture. Hosts need not know how many other members are in the group, nor must they open huge numbers of sockets (Fisher 2002).

- **Addressing of IP multicasting**

  Each multicast distribution tree is represented by a special pseudo-IP address called *a multicast IP address* or *class D address*. A section of the Internet has been set aside for multicast groups (Singhal and Zyda 1999). Moreover, similar to the normal IP addresses, the *multicast IP address* can have more human-readable domain name such as the registered all-routers.mcast.net. A client joins a multicast group by specifying the *multicast IP address*. At present, most operating systems have supported *IP* multicast addressing.

4.4.3 Evaluating and proposing reliable network communication for the Web-based VR system in Net-VEs

According to the examination of Internet protocols described above, this section evaluates network protocols for Net-VEs by comparison of their advantages and limitations and then discusses the selection of protocols for different types of Net-VE architecture design. Furthermore, the section presents a hybrid method of reliable communication by using multiple protocols for Web-based VR systems in Net-VEs.

4.4.3.1 Comparison of network protocols for Net-VEs

It is a difficult task for the Net-VEs designer to select an efficient protocol among the considerable variety of protocols. Each protocol offers its own combination of services and costs, and the protocol choice therefore largely depends on the particular needs of Net-VEs and its underlying communications architecture. Indeed, the Internet protocols examined in Section 4.4.2 (IP Unicasting and IP Multicasting) have been widely used within Net-VEs.

- **Using IP Unicasting (TCP/IP) in Net-VEs**

  Although IP Unicasting protocol (e.g. TCP/IP) provides Net-VE system with reliable data transmission between exactly two hosts, its point-to-point feature limits its use within large-scale Net-VEs because it is difficult to set up and maintain connections between every pair of hosts in Net-VE. In the other hand, the network bandwidth required by the server increases linearly and even exponentially with the number of client hosts. Based on these characteristics, IP Unicasting (TCP/IP) is suitable for small-scale Net-VE systems in server-client architectures.

- **Using IP Multicasting in Net-VEs**

  IP multicasting provides the mechanisms for dealing with immediate data transmission to multiple hosts simultaneously. In addition, multicasting a single update to all members reduces the requirement at any host, whether server or client, to a maximum bandwidth is $N \times P$. Although IP Multicasting is not ubiquitously adopted on the Internet as IP Unicasting now, it will be the most efficient transmission for large-scale Net-VE systems (especially for Collaboration Net-VEs) with large numbers of hosts and heavy data requirements in the near future.

According to **S. Singhal** and **M. Zyda** (Singhal and Zyda 1999), Table 4-2 summarizes the advantages and limitations of the Internet protocols — IP

Unicasting and IP Multicasting and highlights their implication on Net-VEs architecture design.

Network Protocol	IP Unicasting (TCP)	IP Multicasting
Advantages	▪ Reliable data transmission ▪ Transmission flow control ▪ Ubiquitous with Internet	▪ Simultaneous immediate transmission to multiple hosts ▪ Low Bandwidth overhead ▪ Nearly ubiquitous, except efficient Internet-wide delivery
Limitations	▪ Only supports point-to-point connectivity ▪ Bandwidth overhead	▪ No guaranteed reliable packet delivery ▪ Only available for Internet hosts connected to the MBone
For Net-VEs	Small-scale Net-VEs with small number of hosts and limited data requirements, typically used in a client-server architecture	Large-scale Net-VEs with large number of hosts and large data requirements, used in both peer-to-peer and client-to-server architecture, particularly for collaboration applications.

Table 4-2 The Comparison of the network protocols for Net-VEs

4.4.3.2 The hybrid method of reliable communication for the Web-based VR system

For Net-VEs, the reliability of communication is typically the most important requirement — without it, a shared environment will eventually collapse. In order to achieve reliable communication, a hybrid method is emerging by using multiple protocols together within a single Net-VE system. The method intends to take advantage of different protocols' strengths. For example, there is no single reliable multicast protocol because multicasting uses only IP, which does not guarantee data packets delivery nor that packet will arrive in the order they were

transmitted (Bennett and Partridge 1999). The Real-Time Transfer Protocol (RTP) and the Real-Time Transport Control Protocol (RTCP) together provide a solution to address this problem by a feedback or checking mechanism. RTP and RTCP run on top of the User Datagram Protocol (UDP) to provide timing information necessary to synchronize and display multimedia data. The Real Time Streaming Protocol (RTSP) is another protocol relevant to multicasting. RTSP was first published as a proposed standard in April 1998 by the Internet Engineering Task Force (IETF). RTSP acts as a framework for controlling multiple data delivery sessions and assists with switching between TCP, UDP and RTP sessions as needed (RealNetworks 2002). The fundamental difference between TCP/IP and RTSP can be simply stated. A real-time streaming protocol (RTSP) is most concerned about when a data packet arrives while TCP/IP is concerned about if the packet has arrived.

Efforts are also underway to establish protocols to access to the Internet for Net-VEs. Meanwhile, the file formats for supporting streaming media across the Internet have been developed. They include RealMedia from RealNetworks Inc., QuickTime from Apple Computer Corporation, Windows Media Advanced Streaming Format from Microsoft Corporation, and the MetaStream 3D file format from MetaCreations Corporation. Multicasting and RTSP have become important connectivity building blocks for Internetworked Virtual Environments (Carson and Clark 1999). The different communication techniques determine the different level of reliability, functionality, and scalability.

In order to design a cost-effective Net-VE system, it is essential to select the most appropriate protocol for the application requirements. The design tradeoffs are often subtle and complex. In summary, IP Unicasting (TCP/IP) is suitable for small-scale Net-VE systems in server-client architectures. It is the most economical communication used in Net-VEs thanks to the existing Internet-wide connection. IP multicasting is a promising technology for a large-scale Net-VE on the Internet. However, the Net-VEs must provide mechanisms for handling packet loss, data out of order, selecting an agree-upon multicast address for the

application, and discarding packets sent by other applications sharing the same multicasting address. In addition, in the current Internet, not all servers support multicasting, so application designers need to determine whether it is available to all Net-VEs participants. And there is no extra resource and funding to upgrade the existing network communication among SMEs. Thus, in this research project, we have to adopt the common Internet communication — IP Unicasting (TCP/IP) for the proposed Web-based VR system. In the future Internet, with the proposed IP version 6 (IPv6), multicast routing will be built in (Stevens 1998). Just as we take TCP for granted today, tomorrow's designers can assume that multicasting will "just work"— that is, all users will be able to send and receive multicasting messages.

4.5 Conclusions

This chapter described the underlying design and development of the key components — VR database, VR engine, and network communication for the proposed Web-based VR system in Net-VEs to support interactive applications in design and manufacturing. At first, this chapter designed a hybrid method through combining the features of VRML with some advantages of commercial CAD/CAM modelling packages (e.g. Deneb's Virtual NC and QUEST) to develop 3D models of facilities for the creation of the VR database. Secondly, the chapter proposed the generic framework of the VR engine and the model of interactive simulation for the Web-based VR system. It then presented how the capabilities of interactive simulation in VRML were used to develop interactive dynamic behaviour simulations in the proposed Web-based VR system. Finally, the chapter discussed network communication in the design of Web-based VR systems. In particular, it evaluated network protocols by comparison of their advantages and limitations in order to select the most appropriate protocol for different types of Net-VE architecture design. Furthermore, it suggested a hybrid method of reliable communication by using multiple protocols for the proposed system in Net-VEs. The next chapter will describe the details of a specific implementation of the proposed Web-based VR system.

Chapter 5

Implementation

5.1 Introduction

Having developed an overall design for a Web-based VR system, this chapter presents the details of a specific implementation of the architecture. There is therefore a shift of emphasis from consideration of the overall approach and high-level framework to a much more practical implementation.

The chapter presents a specific implementation of a prototype system — VRML-based Factory Layout Simulator (VFLS) to validate the Web-based approach and framework proposed in previous chapters. At first, the overall system functionalities of VFLS are analysed and clarified according to the requirements of factory layout applications. The following sections describe the major implementation components of the prototype system: the construction of Virtual Machine Database (VMD); the creation of grid-enhanced 3D coordinate system; the implementation of "*Inline*" mechanism for the creation of complex VRML scenes; the composition of 3D transformation for layout operations; and the introduction of the '*Virtual Sensors*' for real-time interactive manipulation in a distributed virtual environment through the WWW.

Further, the chapter presents the software development of VFLS using the well-known UML approach of the object-oriented software engineering. The *Use Case* diagram, the *Interaction* diagram, and the *Activity* diagram are integrated to model the software system. The software system has been implemented by the author and the Graphical User Interface (GUI) of VFLS is presented here. As a demonstrator, the implementation of VFLS has successfully validated the viability of the Web-based VR approach and framework proposed in previous chapters.

5.2 Implementation of the VRML-based factory layout simulator (VFLS)

The prototype Web-based VR system allows users to access integrated virtual design and manufacturing environments and share database and information systems via the Internet. Such integrated virtual environments simulate actual design and manufacturing applications with which could be interacted through the WWW. Its primary implementation includes:

- Represent 3D animated facilities in real time.
- Simulate an application in design and manufacturing and evaluate its functions.
- Share and interact with a virtual application visually in real time. The accessible information consists of design and manufacturing data and information such as product specifications, diagrams and descriptions as well as process information.
- Deliver a virtual application across the Internet for distribution and collaboration working.

The section presents a specific implementation of a prototype demonstration system — VRML-based Factory Layout Simulator (VFLS) to validate the Web-based approach and framework proposed in previous chapters. First, the overall system framework and functionalities of VFLS are analysed and clarified

according to the requirements of factory layout applications. The following sections describe the major implementation components in the prototype simulator. The software toolkit of VFLS has been developed by the author using the UML approach of the object-oriented software engineering.

5.2.1 System framework and functionalities

Based on the proposed Web-based VR approach to support design and manufacturing, the distributed interactive VRML-based Factory Layout Simulator (VFLS) as a demonstrator is developed for factory layout applications by the author. It aims to enhance the traditional factory layout applications by taking advantage of interactive information visualization for collaborative e-service.

The simulator will be able to:

- Quickly produce various factory layouts in visual 3D way by manipulating 3D models of machines in virtual environments.

- Be able to manage and cooperate with an extensible VR database — the Virtual Machine Database (VMD).

- Deliver 3D visual factory layouts scenes across the Internet for e-service.

- Allow distributed Web-based users to interactive with the virtual 3D factory layout environments in real time.

- Be compatible with the Web-based VR international standard —VRML.

- Integrate with other Web-based information systems in design and manufacturing for resource sharing.

According to the overall system architecture of the Web-based VR system proposed by the author in Section 3.5, Chapter 3, the functional framework of distributed VRML-based Factory Layout Simulator (VFLS) is developed as shown in Figure 5-1.

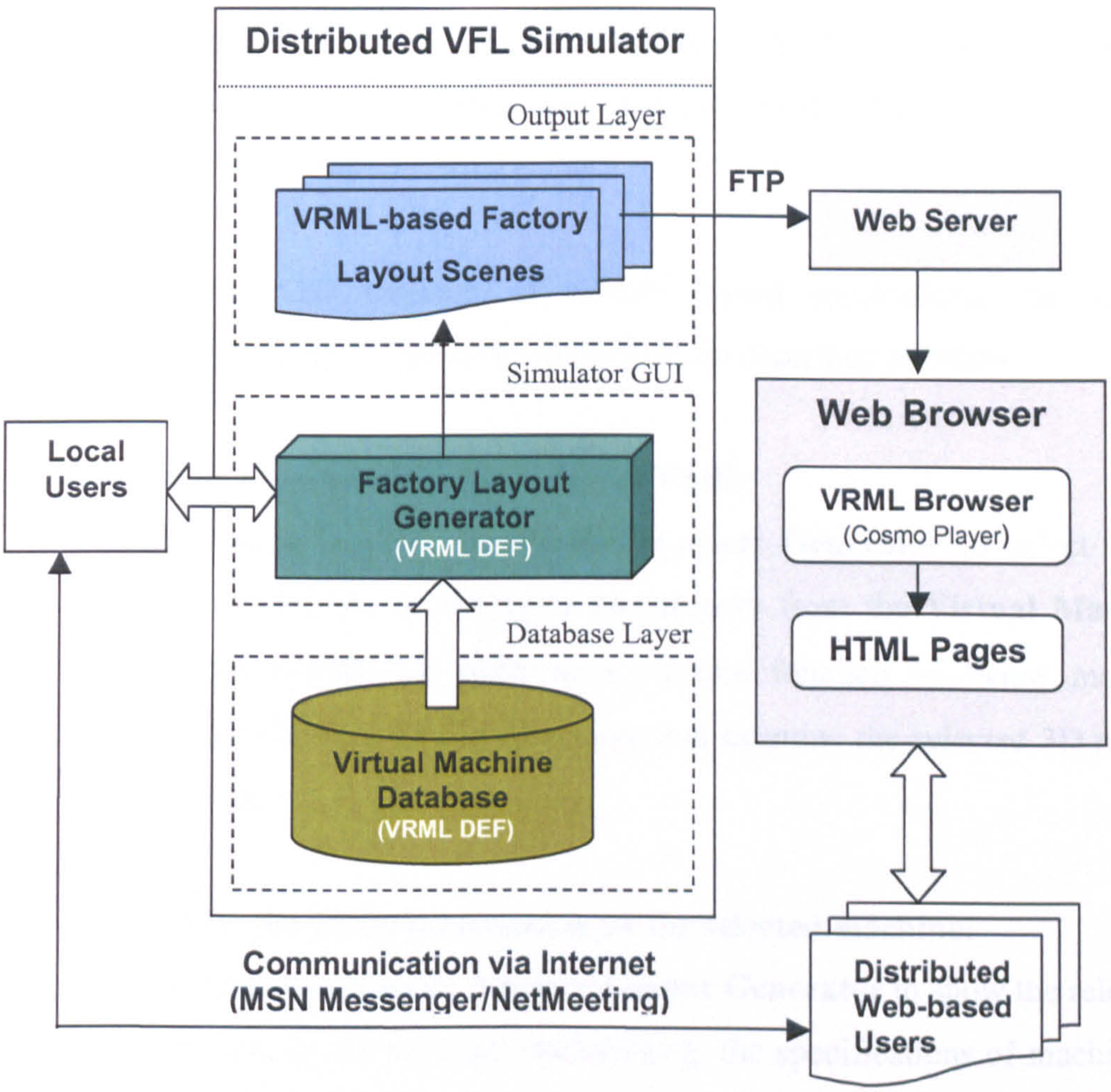


Figure 5-1 The functional framework of the distributed VRML-based Factory Layout Simulator (developed by the author)

The distributed VFLS is made up of three tiers: the database layer, the Graphical User Interface (GUI) layer, and the output layer.

- The database layer includes a VR database — the Virtual Machine Database (VMD) for the factory layout application.

- The GUI layer provides an intuitive, user-friendly interface of the software toolkit — Factory Layout Generator to configure and produce various factory layout scenes in virtual environments.

- The output layer is to organise and save the factory layout scenes in VRML2.0 format, and then upload them into the Web server by a FTP software program.

According to the requirements of factory layout applications, the system functionalities of the prototype software toolkit are described as below.

- **To select machine models from the VMD:**
  The function requires the **Factory Layout Generator** to select a 3D model of machine according to its category from the **Virtual Machine Database (VMD)**. In addition, a relevant function — "view model", which provides the ability to explore and examine the selected 3D model in real-time.

- **To show the useful information on the selected machine:**
  The function requires the **Factory Layout Generator** to show the relevant information on the selected machine (e.g. the specifications of machines). Meanwhile, the file name and path of the selected model are displayed as well.

- **To add and layout machine models into an initial VRML world:**
  The function requires that the **Factory Layout Generator** provides the typical three modes of 3D manipulation — move, scale, and rotation for adding a machine model into an initial virtual environment by its specific input values, including the model's position, size, and orientation.

- **To deliver the factory layout scenes across the Internet:**

  The function requires that the **Factory Layout Generator** is able to save the various factory layout scenes in VRML2.0 format and then upload them onto Web server by a **FTP** program.

- **To communication with distributed Web-based users via the Internet:**

  The function requires that local users can communicate with the other distributed Web-based users by using a peer-to-peer communication software program (e.g. Microsoft MSN Messenger (MSN 2002)) or a Net-meeting program for the group discussion on the various layout scenes.

5.2.2 The construction of Virtual Machine Database

As described in Chapter 3, the VR database is an essential component which provides data for use in Net-VE systems. Thus, the Virtual Machine Database (VMD) as a VR database in the VRML-based Factory Layout Simulator is constructed to integrate, organize, and manage geometric data along with design and manufacturing information associated with 3D models of machines required by factory layout applications. This section presents the implementation of VR database — **Virtual Machine Database (VMD)**.

- ***To obtain geometric data by converting 3D machine models into VRML2.0***

  Obviously, it is cost-efficient to obtain 3D geometric data of machines by integrating the existing CAD packages. As one of critical implementation issues, it has been described in details in Section 4.2.4. In the simulator, the 3D models of machines for factory layout applications have been developed by the author through borrowing from the machine library of Deneb's Virtual NC and QUEST. The original geometric data of a 3D model in Virtual NC can only save as IGES format. Therefore, the IGES

file of a 3D model is converted into VRML 2.0 by a CAD translator – 3D Viewer in this project. As a result, the geometric data of 3D objects are unified into VRML 2.0 format. All machines' VRML models involved in this application are shown as part of the content of VMD in **Chapter 6 — Results and Evaluation.**

- ***To manage and integrate geometric data with relevant information by creating Database Table***

After obtaining the geometric data of 3D objects, the database of VFLS is structured in ***Table*** style. ***Database Table*** allows directly accessing data and information stored in every record and field. In this project, all records of machines in the factory layout application have been stored in the ***Database Table*** — VM.DB. Every data record is constructed as shown in Figure 5-2 to integrate different data and engineering information. It consists of four data fields: ***Category, Model, Description, and Photo.***

a) The ***Category*** field contains the category of a machine such as milling machine.
b) The ***3D Model*** field is responsible for pointing to machine model's VRML files so that the geometric data of 3D machine models can be integrated into the database.
c) The ***Description*** field contains possible design and manufacturing information such as the specification of a machine.
d) The ***Photo*** field links the picture of a machine model.

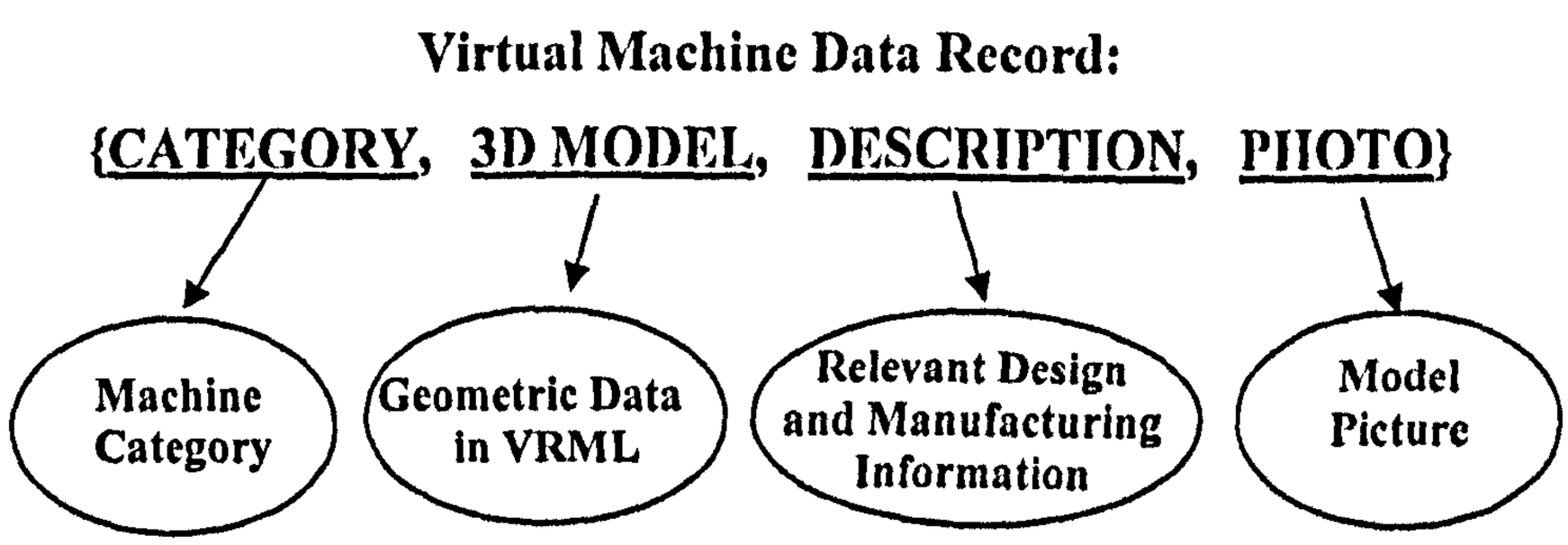


Figure 5-2 The structure of data record in VM.DB

Database Table can be created by different database software packages such as Paradox, dBASE, Access, FoxPro, or an SQL database on a remote server, such as InterBase, Oracle, Sybase, MS-SQL Server, Informix, or DB2. In this system, the database table is created in Paradox by the author and is shown Figure 5-3.

Contents of VM.DB

Definition | Data | Enter SQL

Category	Model	Description	Photo
Category1	Model1-1	(MEMO)	(GRAPHIC)
Category2	Model2-1	(MEMO)	(GRAPHIC)
Category1	Model1-2	(MEMO)	(GRAPHIC)
Category1	Model1-3	(MEMO)	(GRAPHIC)
Category3	Model3-1	(MEMO)	(GRAPHIC)
Category2	Model2-2	(MEMO)	(GRAPHIC)
Category2	Model2-3	(MEMO)	(GRAPHIC)
Category3	Model3-2	(MEMO)	(GRAPHIC)
Category3	Model3-3	(MEMO)	(GRAPHIC)

Figure 5-3 The Database Table of VMD – VM.DB in Borland SQL explorer (Developed by the author)

5.2.3 The creation of grid-enhanced 3D coordinate system

In a VRML-based simulation system, an initial virtual environment is required to be established as a 3D workspace which allows positioning and assembling 3D objects in a scene for factory layout applications. According to the principles and algorithms of three-dimension representation described in (Watt 2000) (Wilson 2000) by convention a 3D space is represented by coordinate systems. The principal coordinate system of a 3D workspace is also known as the global coordinate system or the scene universe. Any point in it can thus be represented by three-axis values (x, y, z) with respect to the origin (0, 0, 0) as shown in Figure

5-4. A grid-enhanced 3D coordinate system has been developed in this project and its implementation procedure is described as below.

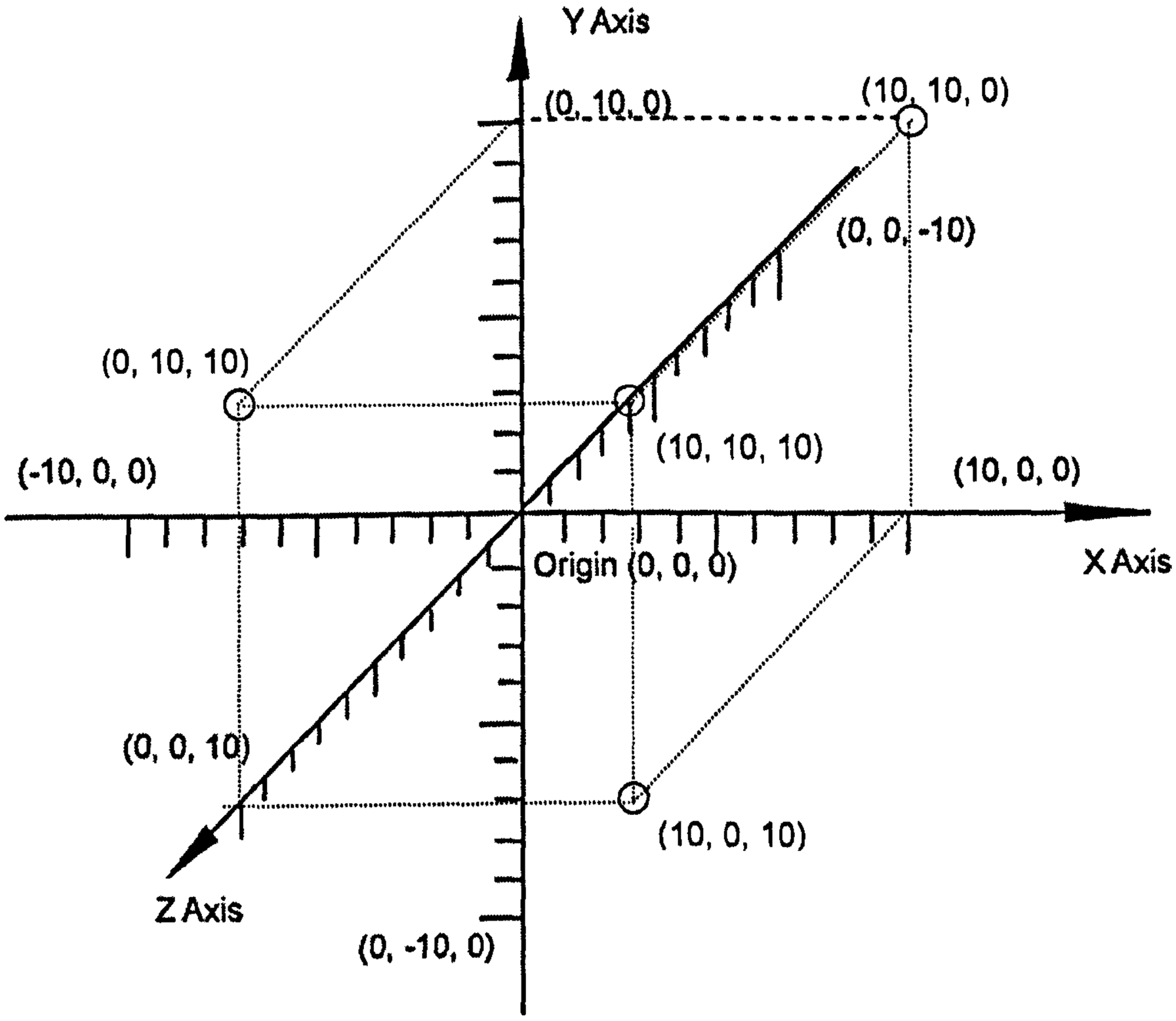


Figure 5-4 Any points in 3D space are represented by three-axis values (x, y, z) with respect to the origin (0, 0, 0) in the coordinate system

- ***To model and assemble 3D axes***

  At fist, each axis is modeled by combining a ***Cone*** node and a ***Cylinder*** node in VRML 2.0. The x, y, z axes are assembled mutually perpendicular and positioned at the origins (0, 0, 0) in the 3D coordinate system and shown in Figure 5-6b.

- ***To generate 3D grid***

  In this project, "*Grids*" have been incorporated into the 3D coordinate system for use in precise positioning, navigation for the location of points in space. The XY-plane Grid, XZ-plane Grid, and YZ-plane Grid have

been individually created in VRML2.0 as shown in Figure 5-5 and finally they are assembled along with axes to fit the 3D coordinate system.

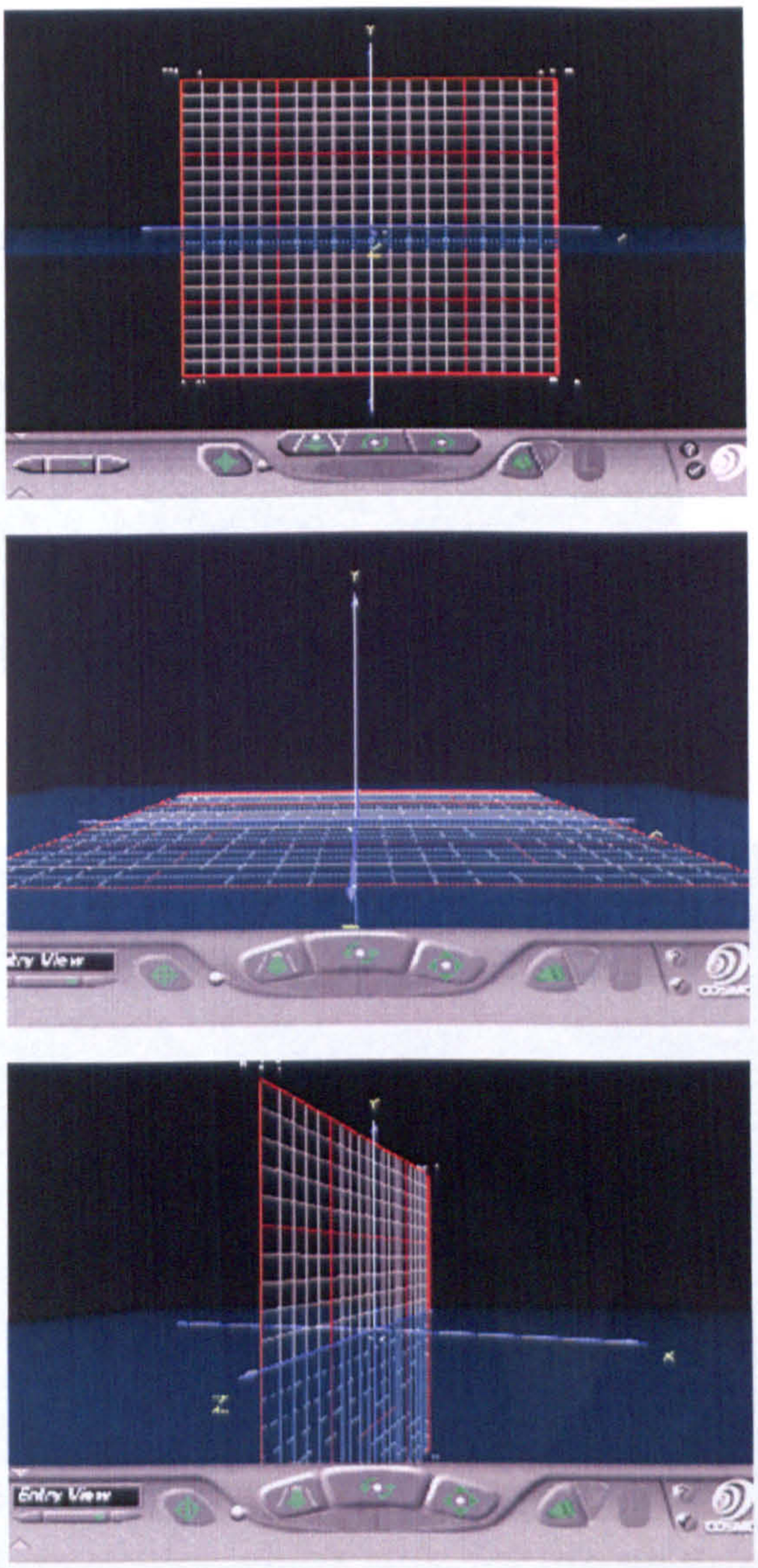


Figure 5-5 The generation of grid-planes — XY, XZ, and YZ.

- ***To make grid-plane moveable***

  In order to enhance the ability of precise positioning in real time, the three "***Grid-planes***" (XY, YZ, and YZ) have been implemented to be able to move along the relevant axis. For an example shown in Figure 5-6a, the plane of XZ-based Grid can move along the Y-axis. With moving up and

down, the precise values of points are changed and are displayed in real time within the 3D space.

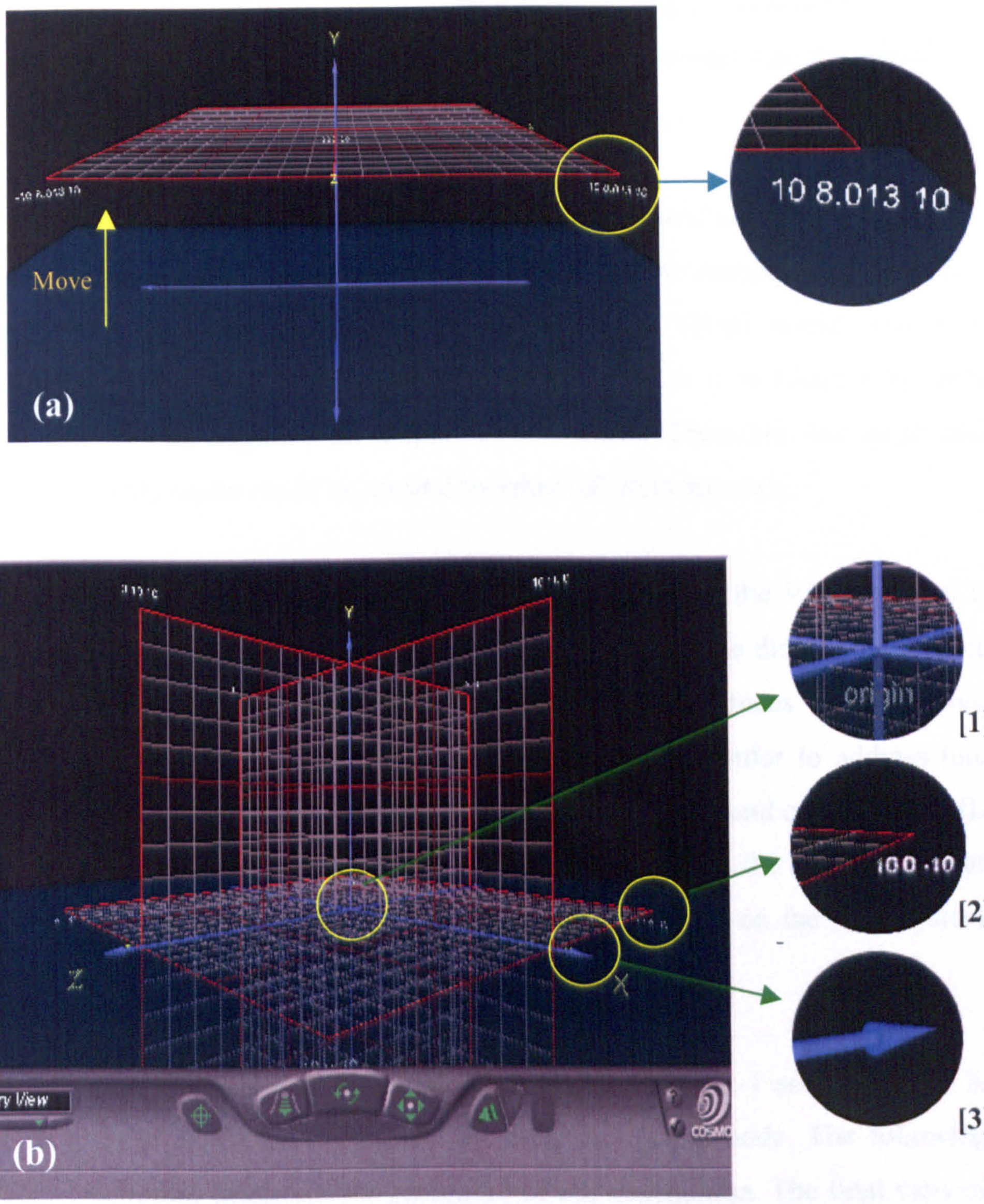


Figure 5-6 The view of the Grid-enhanced 3D coordinate system in VRML. (a) Moving the plane of XZ-based Grid along Y-axis (b) the views of zoom in: [1] Origins [2] the coordinate value for a point [3] 3D representational axis

5.2.4 Inline mechanism for creating complex virtual environments

The more 3D objects are added into a VRML world, the more the complexity of a virtual scene is increased. In the prototype system, adding a machine model into a VRML scene is to integrate the geometric data of the model into the initial 3D workspace.

As the geometric data of a model has been encapsulated in a VRML file, the procedure of adding 3D object into a scene is equivalent to merging its VRML file into the whole scene's VRML file. In addition, since a VRML world exists in a style of parent-children scene graph hierarchical structure (See Chapter 4), each 3D object can be regarded as subpart of the world. Therefore, the large and complex VRML scene could be created by inherited from subparts.

However, in the implementation, it is not practical to insert the VRML file of a 3D object that contains the geometric data into the world file directly because it results in a huge file size of the world file and then reduces its interactive performance after being delivered across on the Web. In order to address this weakness, ***Inline*** mechanism is adopted to implement the big and complex VRML scene in this project. The ***Inline*** node in VRML 2.0 provides the mechanism that allows the inclusion of another VRML file stored anywhere on the local system and even VRML files distributed across the Web.

The following example is to add two 3D machines (Model2-1 and Model1-1 in VMD) into the initial 3D workspace by using the ***Inline*** node. The following VRML description illustrates the procedure of implementation. The final view of VRML scene is shown in Figure 5-7.

```
#VRML V2.0 utf8
#
# VRML-based Factory Layout Simulator
# Authored by Li Jin
# Engineering Design and Simulation Research Group
# School of Engineering and Build Environments
# Contact: jinhull@hotmail.com
}
```

```
Group{
  children[
    WorldInfo{
      title "VRML-based Factory Layout"
      info ["by Li Jin"]
  },
    ...
  ...
}

#end of file
```

The VRML description of initial Grid-enhanced 3D coordinate system

```
#Add new machine
Group{children[Transform{
translation 0 0 0
scale 1 1 1
children Inline{
url "C:\Program Files\Borland\CBuilder5\Wolver\VMLib\Model2-1.wrl"
}
},]}
```

Add a 3D machine –"Model2-1" into the VRML scene at the origin (0, 0, 0)

```
#Add new machine
Group{children[Transform{
translation 3 0 0
scale 1 1 1
children Inline{
url "C:\Program Files\Borland\CBuilder5\Wolver\VMLib\Model1-1.wrl"
}
},]}
```

Add a 3D machine –"Model1-1" into the VRML scene at the position (3, 0, 0)

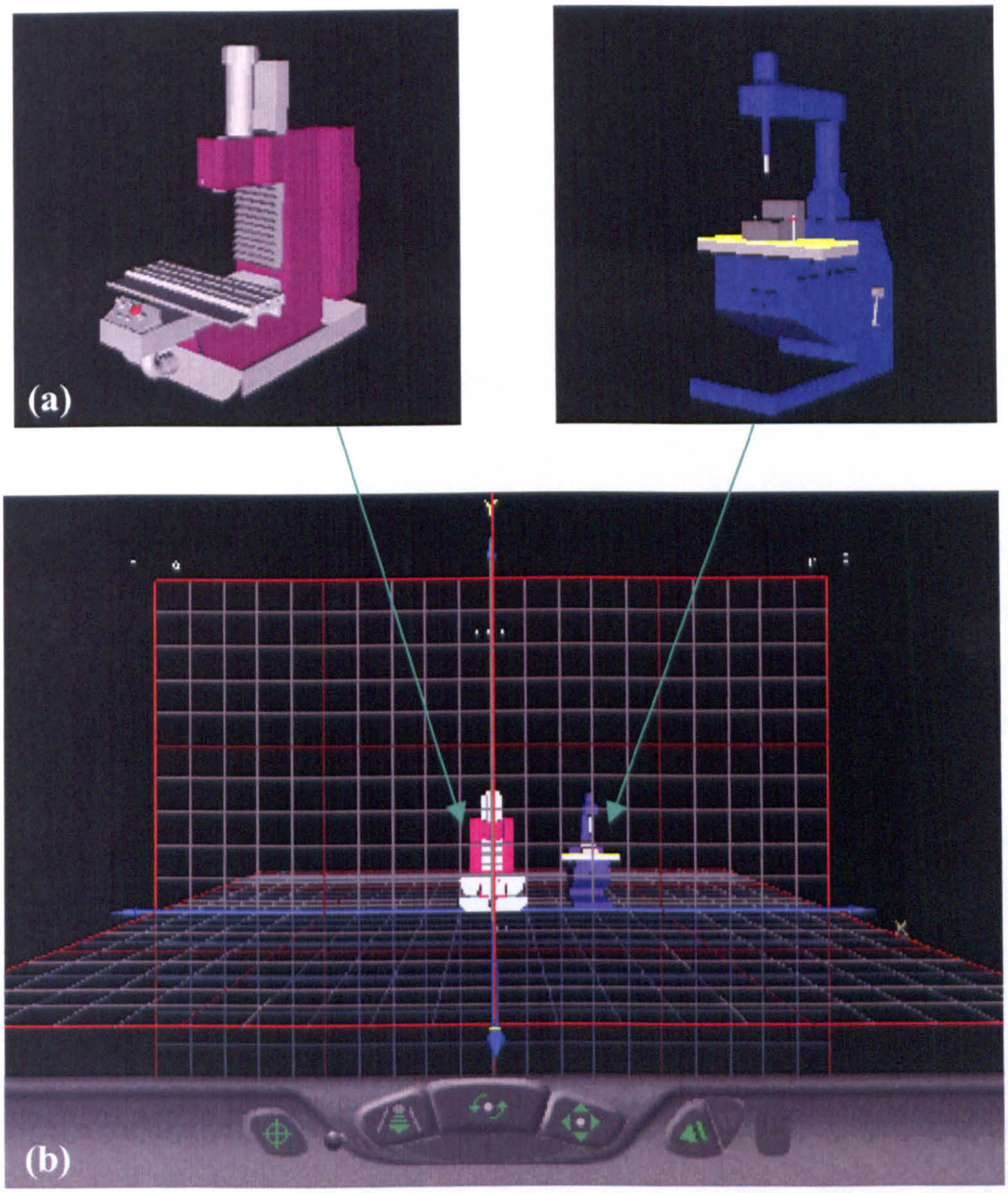


Figure 5-7 (a) The VRML view of 3D virtual machines stored in VMD.
(b) The final view of a VRML scene into which two machines are added by the Inline node (developed by the author)

5.2.5 Composition of 3D transformation for layout operation

General speaking, there are three common attributes in the description of a geometric object layout — position, size, and orientation. They are adjusted and controlled by three layout operations: moving, scaling, and rotation in factory layout applications. Moving is to translate an object's position; Scaling is to change an object's size according to a scaling ratio; Rotation is to orient an abject by a rotation axis and angle. This section presents the implementation of these major functional operations in VFLS by using the composition of 3D transformation in virtual environments.

3D transformations are important tools to transform 3D objects which are usually defined relative to their local coordinate systems into world coordinate systems or the scene universe (Watt 2000). Therefore, they can be used to manipulate objects around in a virtual environment. It is able to effect translation, scaling, and rotation. Technically, a 3D transformation is made up of any combination of translation, scaling, and rotation in order that the relative spatial relationships of objects may be defined. The principles and algorithms of achieving various transformations are presented in (Foley et al. 1997). Generally, 3D transformation can be represented by a matrix operation.

- ***Translation*:** Any points in space can be translated to new positions by adding offsets to their coordinates. The convention in computer graphics is to have the point or vector as a column matrix, preceded by transformation matrix **T**.

The matrix representation of a point (x, y, z) is $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Translation can be represented as a matrix operation: V' = T V

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Implies that object is translated in 3D by applying a displacement tx, ty, tz to each vertex that defines the object. The matrix notation is a convenient way of writing the transformation as a set of three equations:

$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

- ***Scaling***: Any objects in space can be scaled by multiplying a scaling factor to their coordinates. Using matrix notation, scaling can be represented as a matrix operation: V' = S V, **S** is scaling matrix.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The scaling is relative to the origin. Sx, Sy and Sz are scaling factors (for uniform scaling, Sx = Sy = Sz). The process can be expressed by three equations:

$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$
$$z' = z \cdot S_z$$

- ***Rotation***: Any points in 3D space can be rotated around a rotation axis with an angle. Using matrix notation, rotation can be represented as a matrix operation: $V' = R\,V$, R is the rotation matrix.

  When a point is rotated around the coordinate axes: x, y, and z, the rotation matrix respectively are R_x, R_y, and R_z as follows (Vince 2001):

R_x (around the x axis)
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

R_y (around the y axis)
$$\begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

R_z (around the z axis)
$$\begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

When rotating about the x-axis, rotation can be represented as a matrix operation: $V' = R_x\,V$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Algebraically, this is

$$x' = x$$
$$y' = y\cos\alpha - z\sin\alpha$$
$$z' = y\sin\alpha + z\cos\alpha$$

When rotating about the y-axis, rotation can be represented as a matrix operation: $V' = R_y V$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Algebraically, this is

$$x' = x \cos \beta + z \sin \beta$$
$$y' = y$$
$$z' = z \cos \beta - x \sin \beta$$

When rotating about the z-axis, rotation can be represented as a matrix operation: $V' = R_z V$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Algebraically, this is

$$x' = x \cos \gamma - y \sin \gamma$$
$$y' = x \sin \gamma + y \cos \gamma$$
$$z' = z$$

The 3D objects are positioned in a scene by applying various transformations such as translation, scaling or rotation. These transformations are applied to every vertex in the object. An example of 3D transformation as shown in Figure 5-8 illustrates the basic transformation of a 3D object (e.g. a cylinder) defined in a world coordinate system. Figure 5-8(a) and (b) show that a cylinder object is translated from its initial position (0, 0, 0) to its new position (3, 0, 0). Figure 5-8(c) shows the cylinder is scaled at 150% ratio, and (d) illustrates it is rotated about Z axis, the rotation angle is 45°.

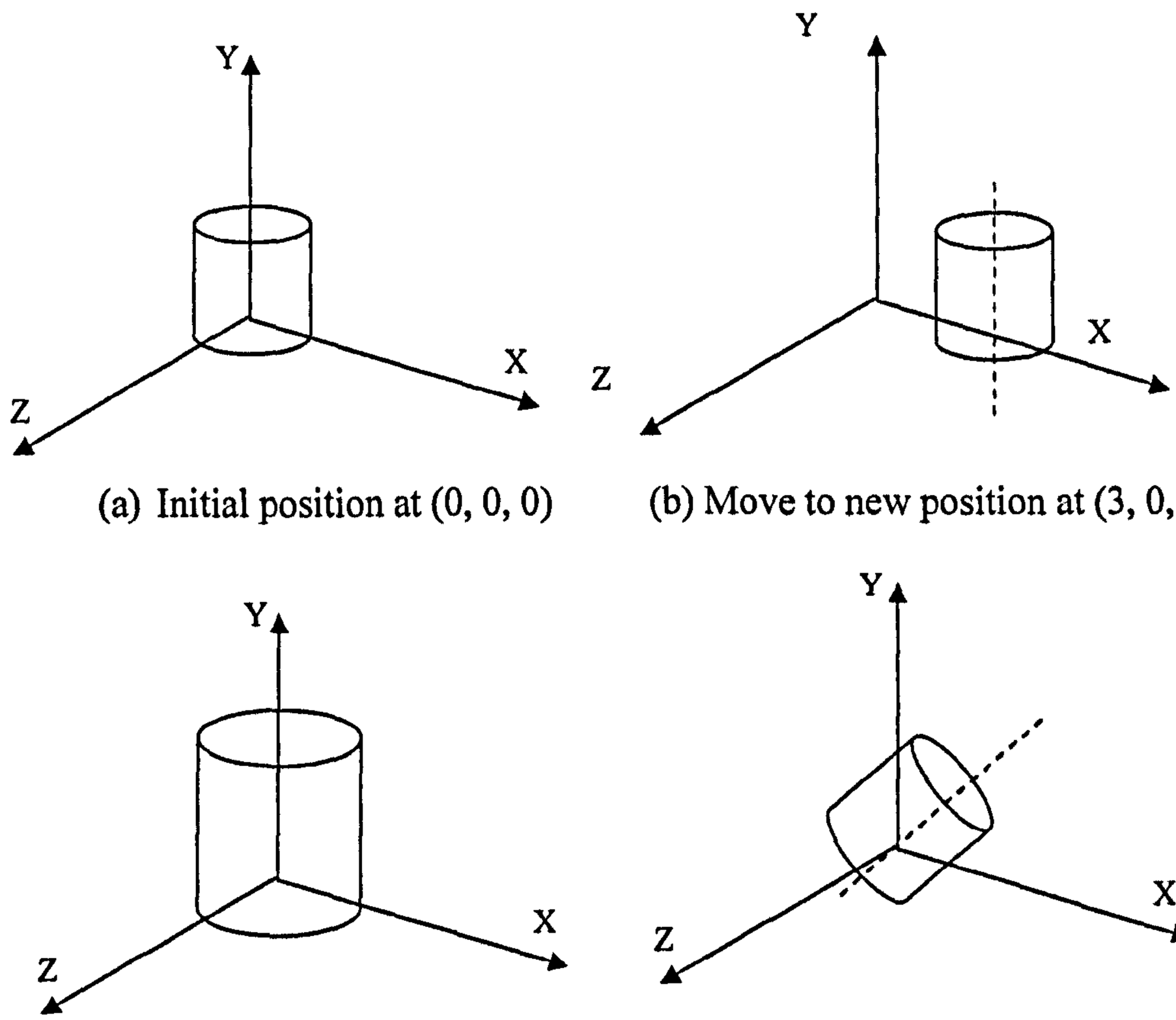


Figure 5-8 An example of 3D transformation. (a) and (b) Translating a cylinder object from its initial position (0, 0, 0) to its new position (3, 0, 0). (c) Scaling the cylinder at 150% ratio. (d) Rotating it about Z axis, the rotation angle is 45°.

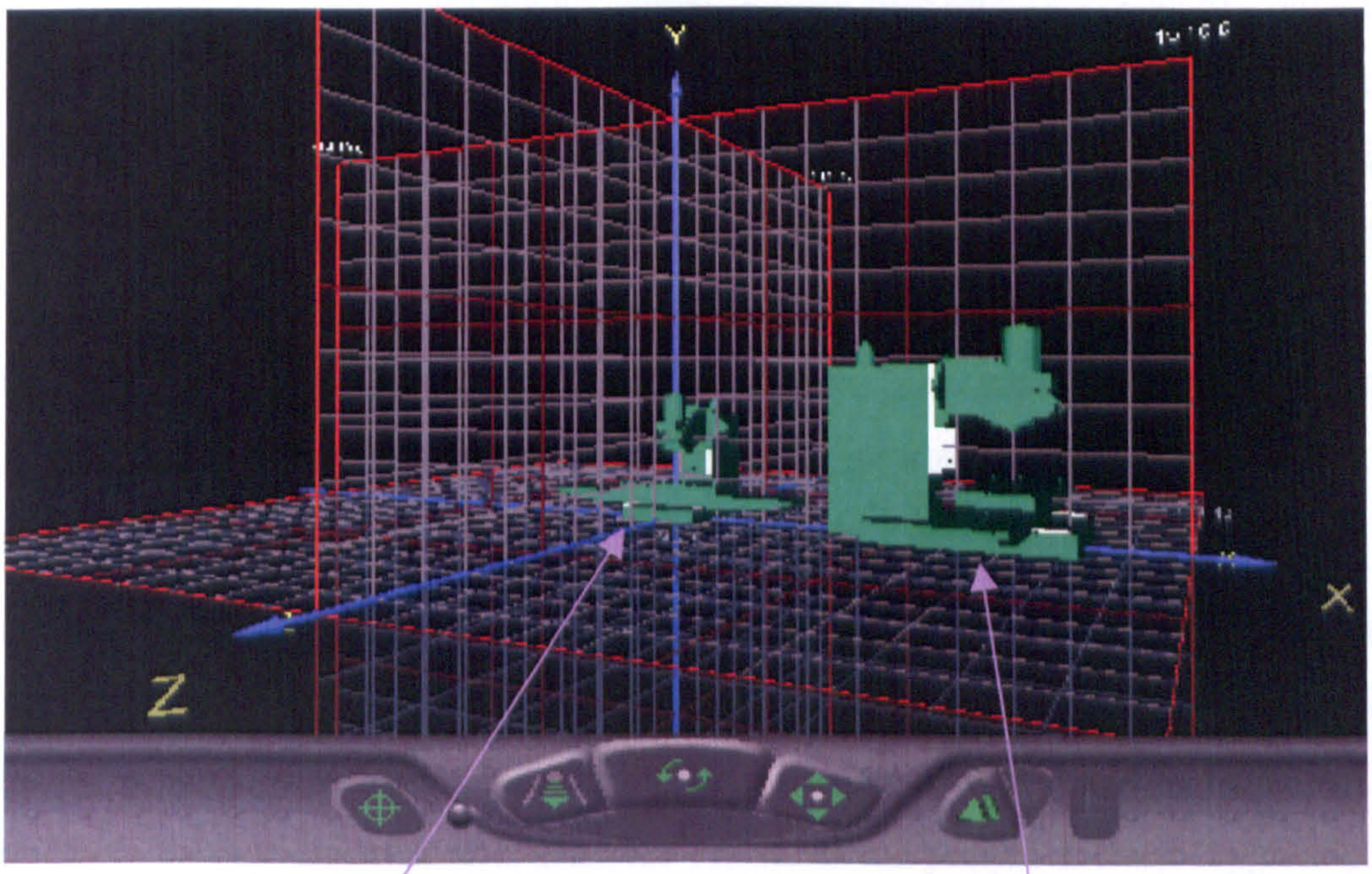
In this project, the composition of 3D transformation has been implemented by the author using **Transform** node in VRML. The **Transform** node is defined in VRML 2.0 as follows:

```
Transform {
  eventIn      MFNode    addChildren
  eventIn      MFNode    removeChildren
  exposedField SFVec3f   center          0 0 0          # (-∞,∞)
  exposedField MFNode    children        []
  exposedField SFVec3f   translation     0 0 0          # (-∞,∞)
  exposedField SFVec3f   scale           1 1 1          # (0,∞)
  exposedField SFRotation rotation       0 0 1 0        # [-1,1],(-∞,∞)
  exposedField SFRotation scaleOrientation 0 0 1 0      # [-1,1],(-∞,∞)
  field        SFVec3f   bboxCenter      0 0 0          # (-∞,∞)
  field        SFVec3f   bboxSize        -1 -1 -1       # (0,∞) or -1,-1,-1
}
```

The ***translation***, ***rotation***, ***scale***, ***scaleOrientation*** and ***center*** fields define a geometric 3D transformation.

- The ***translation*** field specifies a translation to the coordinate system.
- The ***scale*** field specifies a non-uniform scale of the coordinate system. *Scale* values shall be greater than zero. For example, if an object is needed to scale at 50% along *x*-axis, 150% along *y*-axis, and 100% along *z*-axis, it is expressed as "**scale 0.5 1.5 1**". The ***scaleOrientation*** specifies a rotation of the coordinate system before the scaling (to specify scales in arbitrary orientations). The ***scaleOrientation*** applies only to the scale operation.
- The ***rotation*** field specifies a rotation of the coordinate system. For example, if rotation is around *x*-axis and angle is 90°, it is expressed as "**rotation 1 0 0 π/2**".
- The ***center*** field specifies a translation offset from the origin of the local coordinate system (0,0,0).

Thus, layout operations in the VRML-based factory layout simulation system have been implemented in this way to manipulate a 3D object in a virtual layout scene. As shown in Figure 5-9, two identical machine models are added into the initial 3D workspace. One is located at the origin (0, 0, 0) and is kept its original size (scale 1 1 1) without any rotation (rotation 0 1 0 0). Another is translated to the position (7, 0, 0) (translation 7 0 0) and is scaled at 150% ratio (scale 1.5 1.5 1.5) and then is rotated 90° about *y*-axis (rotation 0 1 0 π/2).



VRML Description	VRML Description
translation 0 0 0 scale 1 1 1 rotation 0 1 0 0	translation 7 0 0 scale 1.5 1.5 1.5 rotation 0 1 0 1.5707963

Figure 5-9 The implementation of layout operations in VFLS by the composition of 3D transformation in VRML (developed by the author).

5.2.6 Virtual Sensors for real-time interactive manipulation

In this project, the prototyped VRML-based Simulator aims to use Web-based VR technology to support design and manufacturing applications. In comparison to traditional CAD/CAM systems, such a VRML-based simulator has a novel feature — real-time interaction, which allows the user to interact with the virtual environment in real time. The VFLS is able to produce various factory layout scenes for distributed users on the Web. After the VRML-based scenes are delivered on the web, these layout scenes require providing a means of interaction to allow distributed users to manipulate the 3D objects in the virtual environment in real time. In this project, the "*Virtual Sensor*" as a means of real-time interaction has been introduced by the author into the VRML-based Factory Layout Simulator in order to implement real-time interactive manipulation for layout applications. This section describes the implementation of virtual sensors in VFLS by taking advantage of pointing-device sensors defined in VRML 2.0.

In VRML, *Drag sensors* are a subset of pointing-device sensors. There are three types of drag sensors: *CylinderSensor*, *PlaneSensor*, and *SphereSensor*. The pointing device controls a pointer in the virtual world. While activated by the pointing device, a sensor will generate events as the pointer moves. Typically the pointing device may be categorized as either 2D (e.g., conventional mouse) or 3D (e.g., wand).

- The *CylinderSensor* node maps pointer motion (e.g., a mouse or wand) into a rotation on an invisible cylinder that is aligned with the Y-axis of the local coordinate system. Thus, it can be used to rotate a 3D object around its Y-axis in real-time interaction.

- The *PlaneSensor* node maps pointing device motion into two-dimensional translation in a plane parallel to the Z=0 plane of the local coordinate system. Thus, it can be used to move a 3D object in XY plane in real-time interaction.

- The *SphereSensor* node maps pointing device motion into spherical rotation about the origin of the local coordinate system. Thus, it can be used to implement spherical rotation of a 3D object in real-time interaction.

In this project, the *PlaneSensor* node is adopted in VFLS to allow distributed users on the Web to real-time interactively move a machine model in the pre-defined area of XZ plane (ground plane) for layout simulation. For the example shown in Figure 5-10, a virtual sensor is bound to the machine model for real-time moving manipulation in factory layout applications. It allows the user to drag and move the machine to any position in pre-defined area on the ground (XZ plane) in real-time. It is implemented in VRML 2.0 by the author as follows.

```
#VRML V2.0 utf8
#Authored by Li Jin
#Title: virtual sensor for real-time interactive manipulation

Transform{
translation 0 0 0
rotation    1 0 0 -1.57

children[

DEF movesensor PlaneSensor {
      minPosition -10 -10
      maxPosition 10  10
      }

DEF machine Transform{
      rotation 1 0 0 1.57
      children[

      DEF object Transform {
            scale 4 4 4
            children [Inline {url [ "machine2-1.wrl"]} ]
            }
      ]
}
] }

ROUTE movesensor.translation_changed TO machine.set_translation
```

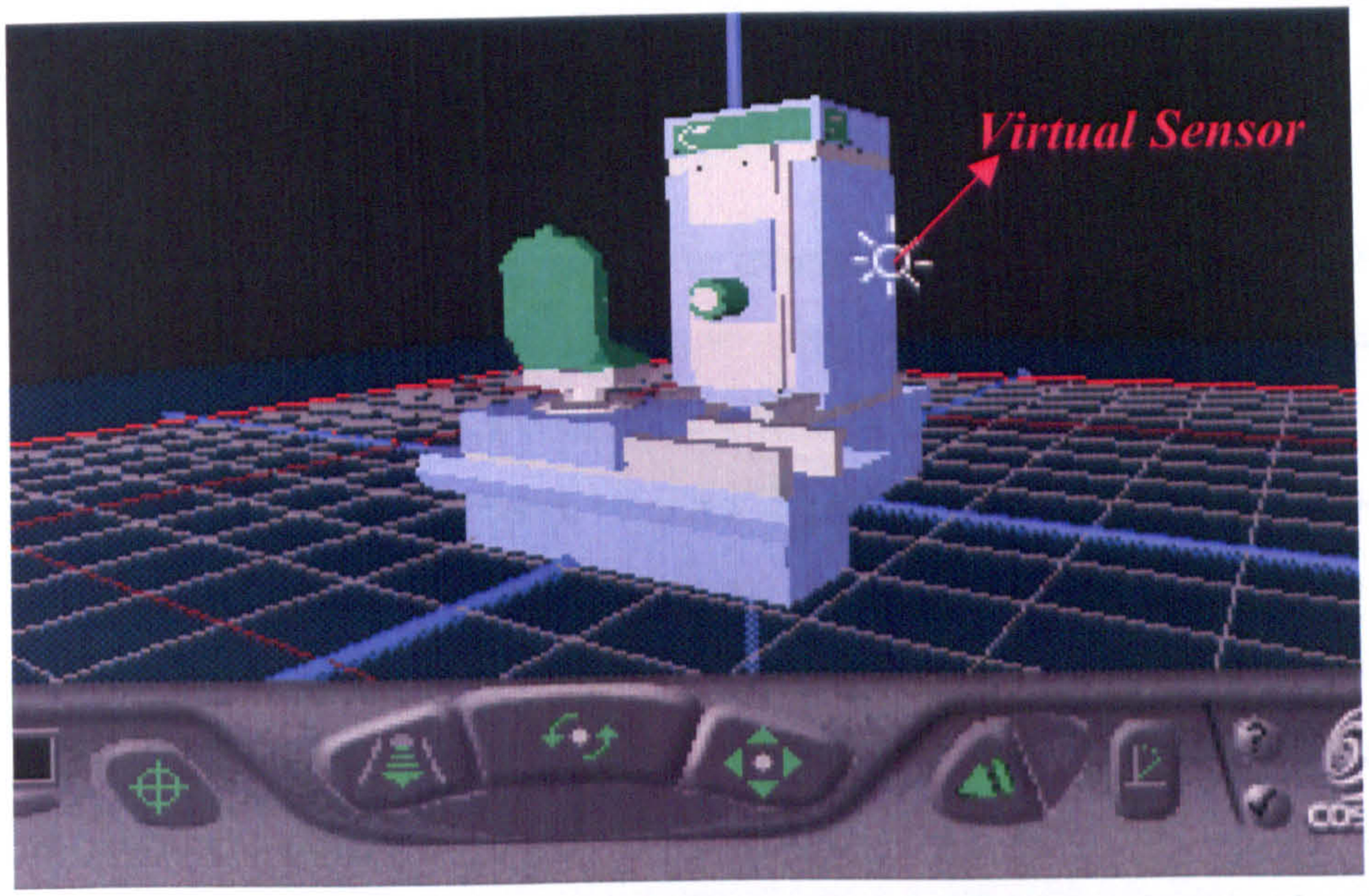


Figure 5-10 Drag the virtual sensor to move a machine around a pre-defined area on the ground -XZ plane (developed by the author)

5.3 UML approach for software development

Based on the analysis of overall functionalities of the distributed VRML-based Factory Layout Simulator described in Section 5.2.1, the Unified Modelling Language (UML) approach — the industry-standard modelling language in software engineering has been adopted for specifying, visualizing, constructing, and documenting the software package of the simulator in this research project.

As a common object-oriented modelling language, UML supports the modelling of complex application specific object-structures. It focuses on the phases of a software life cycle like object-oriented analysis and design before coding (Booch, Rumbaugh, and Jacobson 1999). Using a UML-based model, those responsible for a software development project's success can assure themselves that system functionality is complete and correct, end-user needs are met, and program design supports requirements for scalability, robustness, security, and extendibility, before implementation in code. In addition, UML can model any type of applications, running on any type and combination of hardware, operating system, and network. It is a natural fit for object-oriented languages and environment such as C++, Java, and the recent C#.

The UML approach uses different diagrams to give different views on a software system (Henderson-Sellers and Unhelkar 2000). In this project, the Use Case diagram, the Interaction diagram, and the Activity Diagram have been integrated to model the software system during the development of the VFLS. These UML diagrams are designed by the author and shown in this section.

5.3.1 Use Case design

In software engineering, Use Case Diagram in the UML approach is often used to describe the functionality offered by a software system to a user. The Use Case diagram focuses on functionality rather than data that can lead the use case user to deviate away from an object-oriented analysis. In order to design and develop the

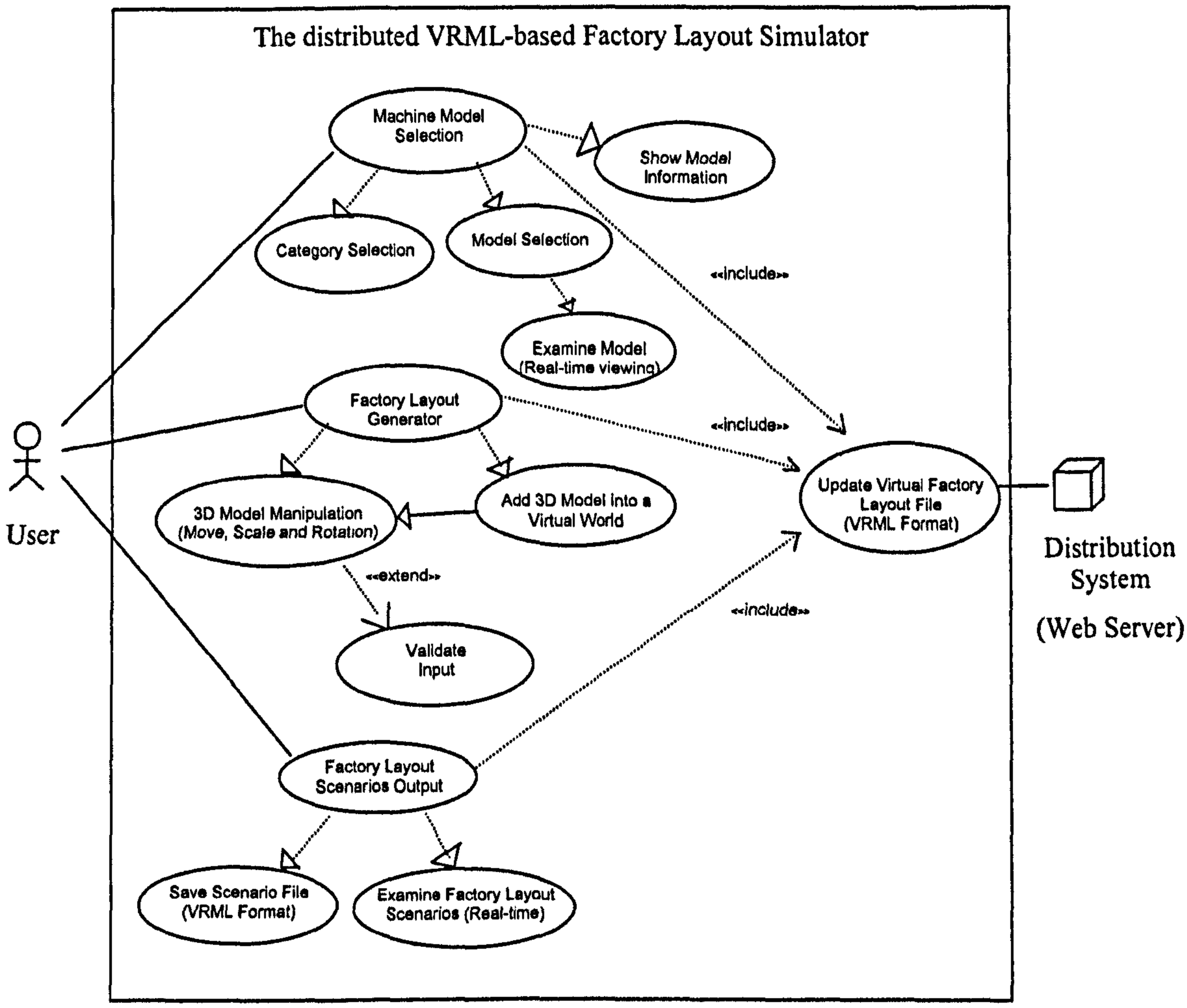
software package of the VFLS, firstly, the Use Case diagram is designed to represent the functional view of the software system. Figure 5-11 shows all system functionalities involved in the simulator. The Use Case diagram is important for visualizing, specifying, and documenting the behaviour of each software element. It makes systems, subsystems, and classes approachable and understandable by presenting an outside view of how those software elements may be used in context.

5.3.2 Sequence Diagram design

The Sequence Diagram is a type of Interaction Diagrams that models the dynamic aspect of the whole software system. It emphasizes the organization of the objects that participate in an interaction. Figure 5-12 shows the Sequence Diagram design of the VFLS. Sequence Diagrams describe interactions among objects in terms of an exchange of messages over time. Each object is rearranged in a linear fashion across the top of the page and a time line for that object is drawn vertically downwards. Horizontal arrows are drawn for each message passing. Different arrowhead styles are available to denote synchronous and asynchronous messages if desired. Thickening of a temporal line indicates when the individual object is active (i.e. the object is receiving a message, executing an operation or sending a message). An arrow going directly to an object box indicated the actual object creation caused by that incoming message. Object destruction is shown by a large cross. Internal message sends are shown as arrows turning back to the lifeline; taking a finite amount of time to execute.

5.3.3 Activity Diagram design

Activity Diagrams are used to model the dynamic aspect of the workflow and behaviours, and the core functions inside the system. Figure 5-13 shows the Activity Diagram design of the VFLS. It illustrates the dynamic nature of this software system by modelling the flow of control from activity to activity. An activity represents an operation of functionalities in the VFLS.



Note:

Generalization ——▷ between actors or between use cases, linked together by a generalization relationship.

Realization ·····▷ between use cases, dependency relationships: <<include>> and << extend>>

- <<include>> relationship permits a use case to represent functionality that might need to be included in more than one use case: it represents a mandatory situation.
- <<extend>> relationship describes how a use case representing additional functionality often describing an anomalous situation or an exception, beyond that provide in the base use case.

Association ———— between actors and use cases only.

Actor (Human role external)

Node

Figure 5-11 The Use Case Diagram design of the VFLS software system

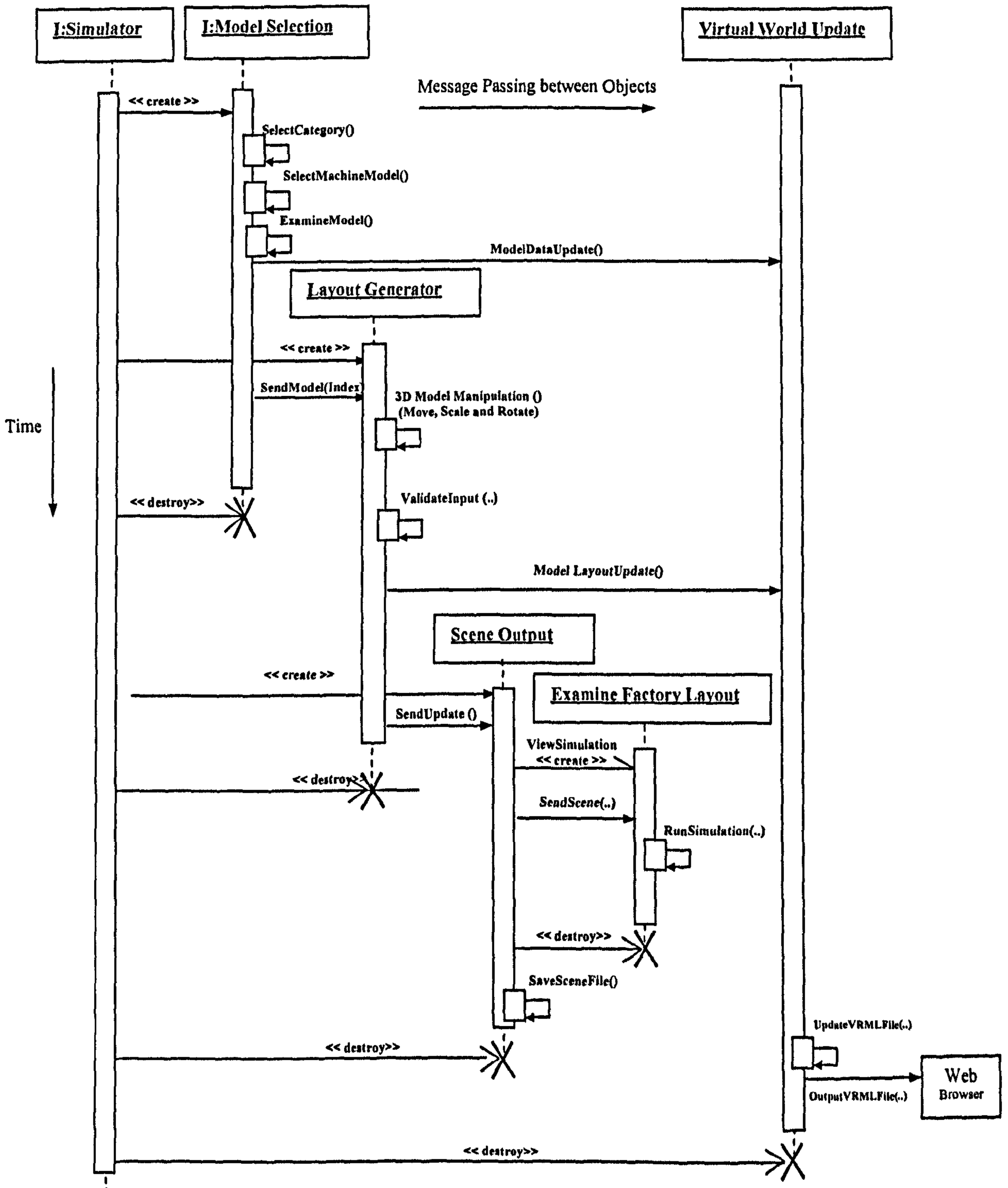


Figure 5-12 The Sequence Diagram design of the VFLS software system

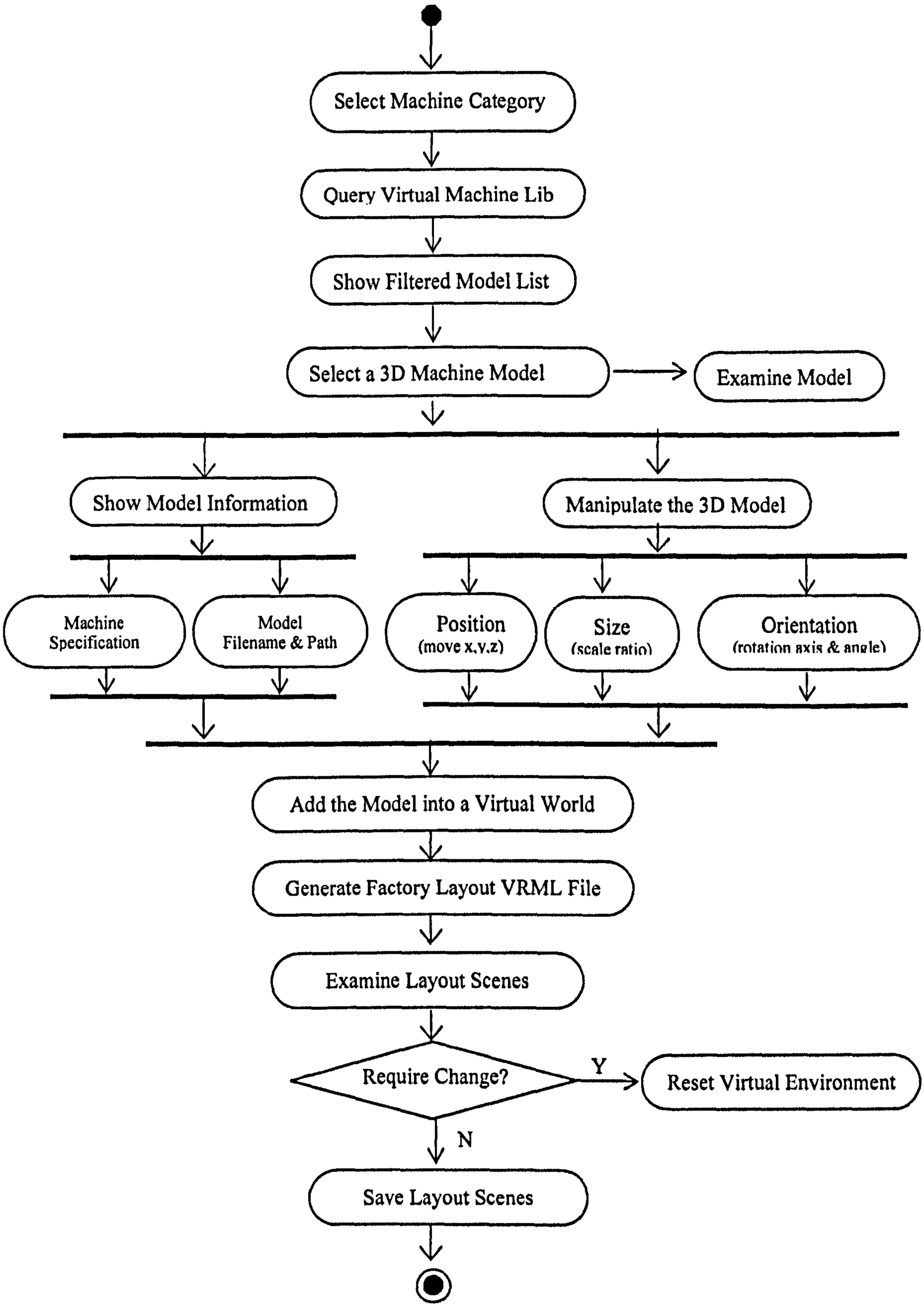


Figure 5-13 The Activity Diagram design of the VFLS software system

5.3.4 Graphical user interface (GUI) of VFLS

As VRML does not define an Application Programmer Interface (API), VRML files are usually coded in text editors like Notepad or WordPad program. Such non-visual programming hinders software developers from creating, processing, and modifying a large and complex virtual scenario efficiently in VRML format. Especially, such weakness restricts engineers in SMEs, who are non-VRML specialists, to get benefits from such Web-based VR technology. Arising from the constraint, the VFLS requires a Graphical User Interface (GUI) that allows the user to produce various factory layout scenes visually and efficiently.

A well-designed Graphical User Interface (GUI) for a software system can greatly increase its usability. According to the principles and techniques of GUI design described in (Constantine and Lockwood 1993), (Weinschenk et al 1997), and (Galitz 2002), the GUI of the VFLS software toolkit is designed to be intuitive, effective, user-friendly in order to support the overall system functionalities on basis of the psychology and physiology involved in the Human Computer Interaction (HCI). The GUI has been implemented by the author in Windows 2000 style and its widgets are shown in Figure 5-14.

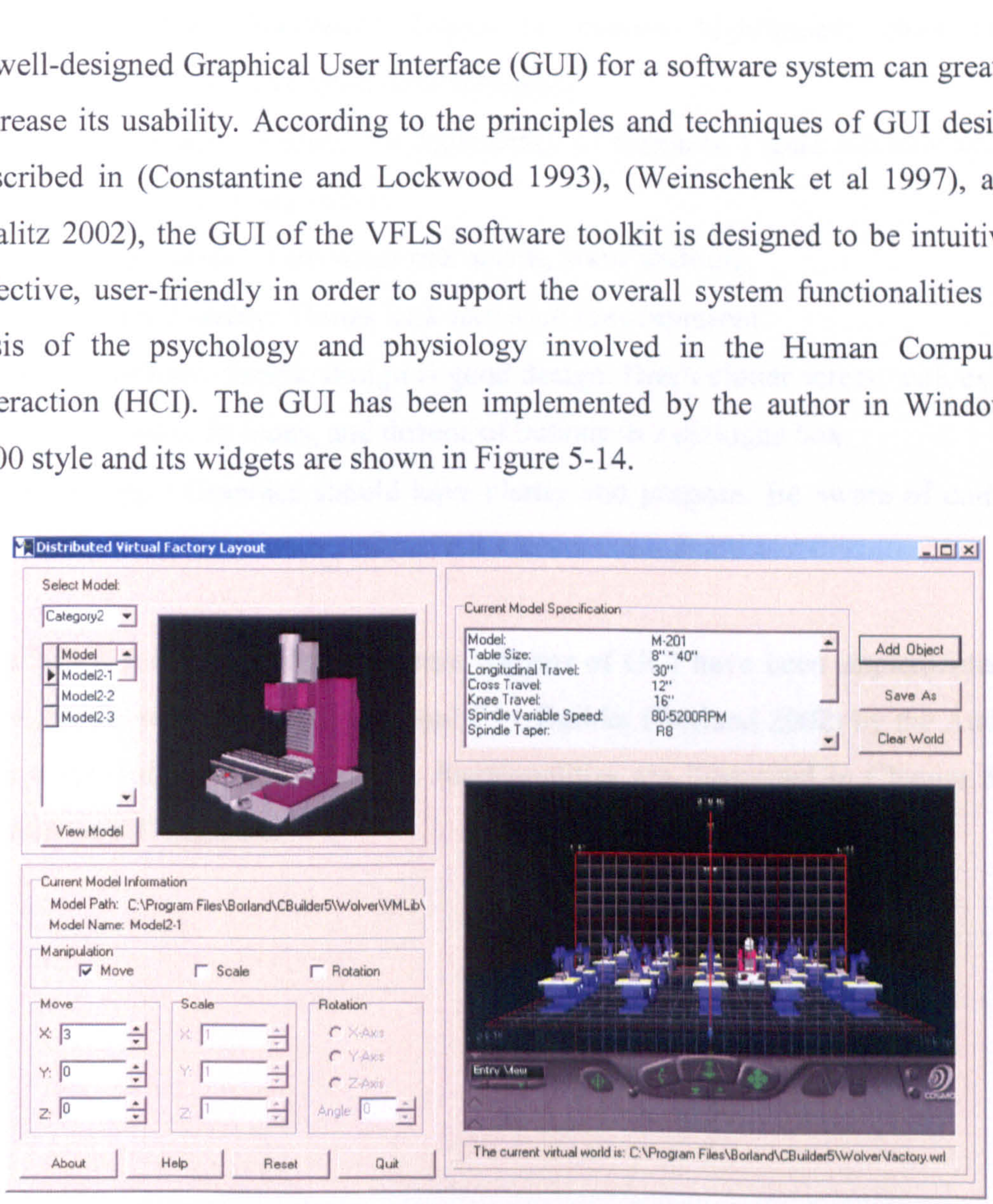


Figure 5-14 The GUI of VFLS software toolkit (developed by the author)

It consists of *widgets* that allow the user to interact with the software application. Widgets include icons, menus, buttons, selection boxes, scroll bars, and other design elements. It is intended to match *Human Interface Design Guidelines* (Apple Computer 1986) as below.

- ***User-controlled:*** hierarchical dynamic windows, control panel.
- ***Friendly Dialogue:*** Plain English, no jargon.
- ***Consistency:*** Applications should share interfaces.
- ***Direct Manipulation:*** Expectation that physical actions have physical results.
- ***Immediate Feedback:*** Topics of interest highlighted, clues about functioning. Give good error messages.
- ***Exploration:*** Reduce the dependency of manuals. Figure out how system works by playing with it.
- ***Forgiveness:*** Warn when risk action, allow undoing.
- ***Visual Fidelity:*** Things look like what they represent.
- ***Simplicity:*** Simple design is good design. Don't clutter screen with excess windows, 20 icons, and dozens of buttons in a dialogue box.
- ***Clarity:*** Graphics should have clarity and purpose. Be aware of cultural differences.

The VFLS software toolkit and these widgets of GUI have been implemented in C++ and developed by using Borland C++ Builder (Borland 2002) by the Author. The results of the major software functionalities are illustrated in Chapter 6 — ***Results and Evaluation***.

5.4 Conclusions

Based on the Web-based VR approach and framework proposed by the author in previous chapters, the chapter described a specific implementation of a prototype system for factory layout applications — the VRML-based Factory Layout Simulator (VFLS). The major implementation components of the prototype system include: the construction of Virtual Machine Database (VMD); the creation of a grid-enhanced 3D coordinate system; the implementation of using the "*Inline*" mechanism for the creation of complex VRML scenes; the composition of 3D transformation for layout operations; and the introduction of the '*Virtual Sensors*' for real-time interactive manipulation in a distributed virtual environment such as the WWW.

Further, the chapter presented the software development of VFLS by using the well-known UML approach of object-oriented software engineering. The development integrated the Use Case diagram, the Interaction diagram, and the Activity Diagram together to model the software system. The Graphical User Interface (GUI) of VFLS was designed and developed to be intuitive, effective, and user-friendly in order to support the overall system functionalities on basis of the psychology and physiology involved in the Human Computer Interaction (HCI) in this research project.

Therefore, as a demonstrator, the implementation of VFLS has successfully validated the viability of the Web-based VR approach and system proposed in previous chapters.

Chapter 6

Results and Evaluation

6.1 Introduction

The previous chapter described the development of the prototype interactive VRML-based Factory Layout Simulation (VFLS) system. The VFLS has been implemented by the author to demonstrate the proposed Web-based Virtual Reality approach and framework to support design and manufacturing via the Internet for collaborative e-service in this research.

This chapter firstly presents the software toolkit — VRML-based Factory Layout Simulator as a demonstrator and describes the results in terms of its functionality and performance. The following sections present the content of the VR Database — Virtual Machine Database (VMD) created in the simulator; show various factory layout scenes produced in this interactive simulator as examples to validate the usability of the software toolkit; and then describe the result of a collaboration working experiment which has shown that the VFLS developed by the author can efficiently support multiple users for collaboration working in design and manufacturing over a local or wide area network.

In order to achieve the final objective of the research project which allows the Web-based user to get benefits in a distributed virtual environment, the chapter

describes the final distribution of the VRML scenes across the Internet and the integration of HTML-based information systems through the Web. In addition, the chapter carries out the evaluations on the VFLS in terms of its database — VMD and the interaction performance and the overall evaluation of the generic Web-based VR system proposed in this project.

6.2 Interactive VRML-based Factory Layout Simulator

Based on UML approach of the software system presented in Chapter 5, the VFLS software system and its GUI were programmed in C++ and developed by the author using Borland C++ Builder (Borland 2002). The user manual and tutorial of VFLS software toolkit and its source code are attached in Appendix A and B. This section presents results in terms of its functionality and performance of the prototype. The results of major system functionalities are illustrated as shown in Figure 6-1.

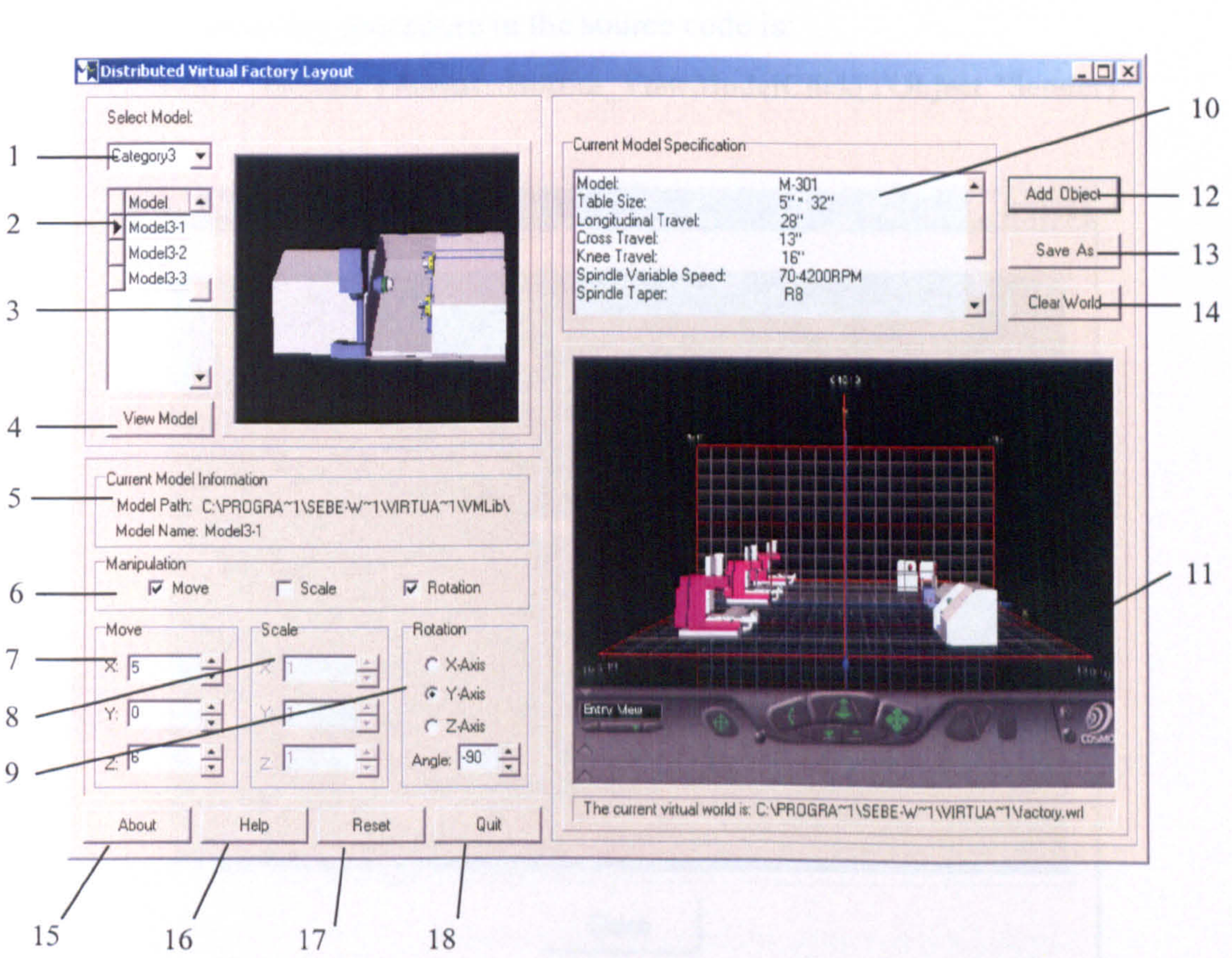


Figure 6-1 The VFLS Software Toolkit (developed by the author)

1. ***Select Object's Category:*** This function allows selecting a category of objects from database. The result is implemented by the following procedure in source code:

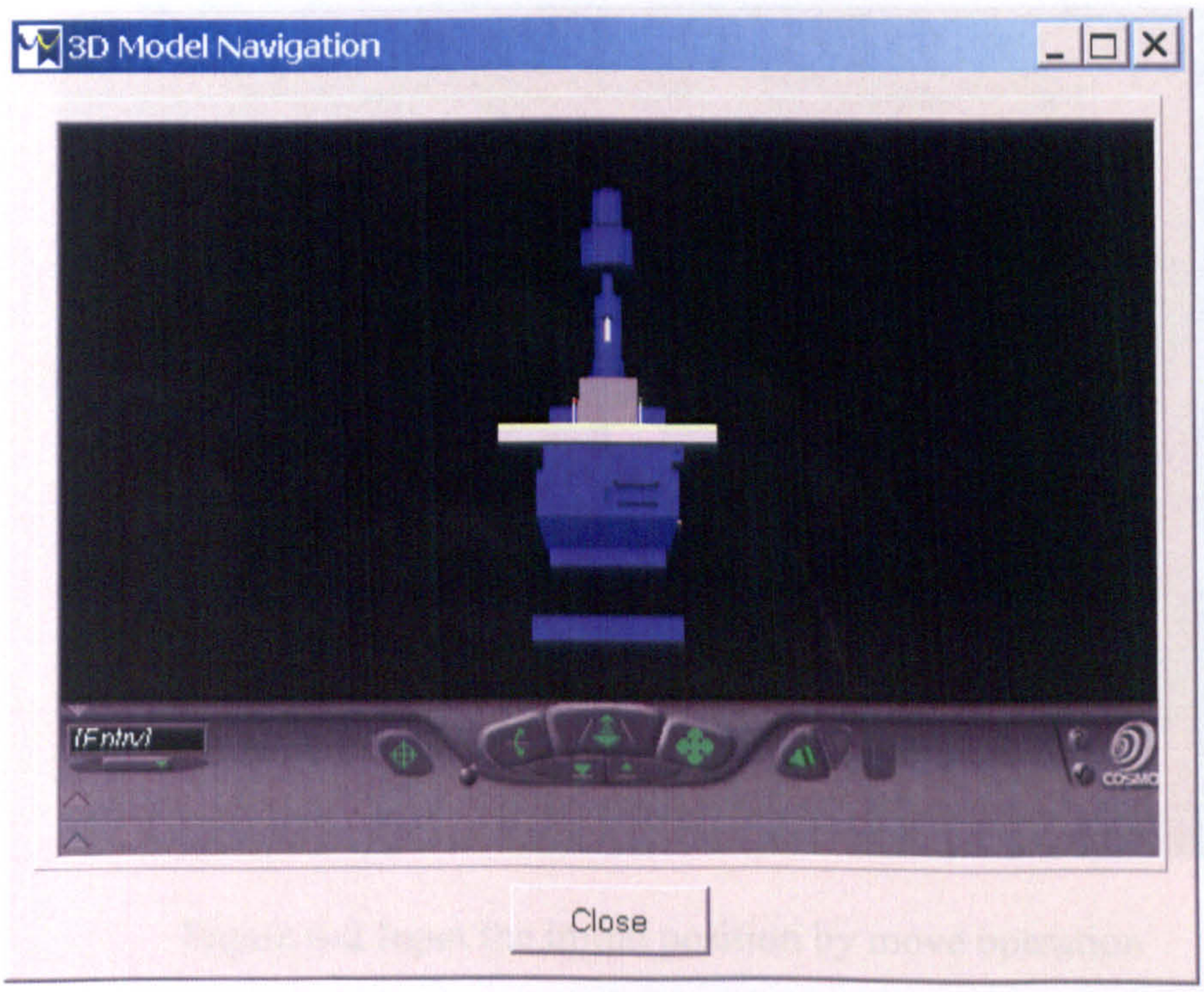
   void __fastcall TForm1::ComboBox1Change(TObject *Sender)

2. ***List Object's Filename:*** This function lists the filenames of objects belonged to the selected category stored in VMD.

3. ***Show Object's Image:*** This function shows the picture of the selected objects stored in VMD.

4. ***View Object in Real Time:*** This function allows exploring and viewing the selected object's 3D model in real-time. Click on the "View Model" button, the following window comes up. The embedded VRML browser is able to navigate the selected model from any viewpoints. The corresponding procedure in the source code is:

   void __fastcall TForm1::Button_ViewModelClick(TObject *Sender)



5. ***Show Current Object Information:*** This function displays the path and filename of the selected object.

6. ***Choose 3D Manipulation for Layout Operation:*** This function allows switch on/off the 3D manipulation — *move*, *scale*, and *rotation* for layout operations. They are implemented by the following procedures respectively:

   void __fastcall TForm1::CheckBox_MoveClick(TObject *Sender)

   void __fastcall TForm1::CheckBox_ScaleClick(TObject *Sender)

   void __fastcall TForm1::CheckBox_RotationClick(TObject *Sender)

7. ***Move Operation:*** This function allows the user to input an initial position value (x, y, z) for the selected object. The result is implemented by the following procedures:

   void __fastcall TForm1::Edit_MXChange(TObject *Sender)

   void __fastcall TForm1::Edit_MYChange(TObject *Sender)

   void __fastcall TForm1::Edit_MZChange(TObject *Sender)

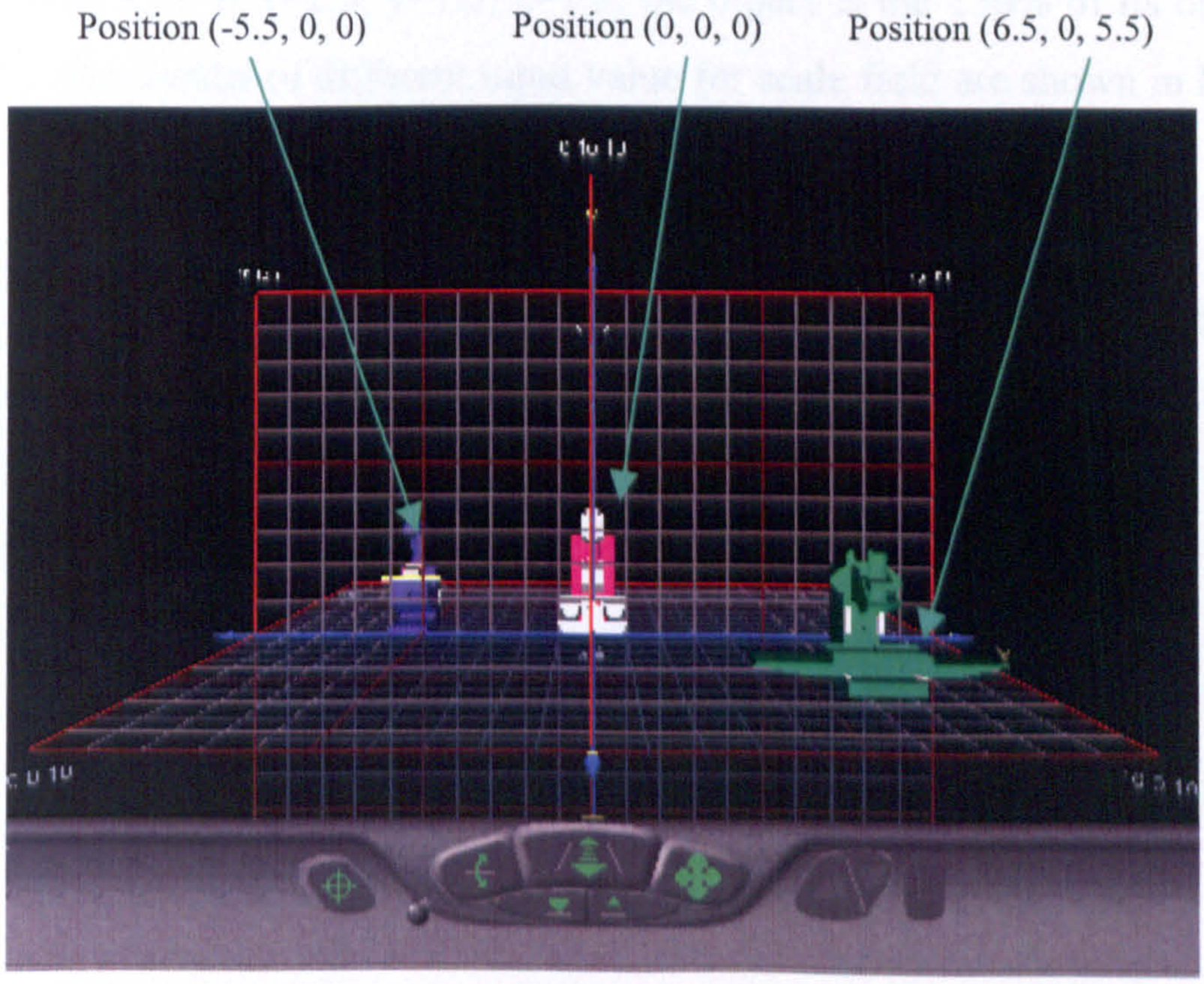


Figure 6-2 Input the initial position by move operation

For example shown in Figure 6-2, three machines are positioned in the virtual environment. Their initial position values input from the user are (-5.5, 0, 0) for Model1-1, (0, 0, 0) for Model2-1, and (6.5, 0, 5.5) for Model2-2.

8. ***Scale Operation:*** This function allows the user to input a scaling ratio for the selected object. The result is implemented by the following procedures:

```
void __fastcall TForm1::Edit_SXChange(TObject *Sender)
void __fastcall TForm1::Edit_SYChange(TObject *Sender)
void __fastcall TForm1::Edit_SZChange(TObject *Sender)
```

The scaling ratio should be larger than zero. If x=1, y=1, z=1, the object is the original size. If x=0.3, y=0.3, z=0.3, the object is the 30% of its original size. If x=1.5, y=1.5, z=1.5, the object is the 150% of its original size. The results of different input value for scale field are shown in Figure 6-3a.

In addition, it supports a non-uniform scaling as well. For example, as shown in Figure 6-3b, if the scale input values are (1.5, 1, 1), the object only enlarges 50% along its X-axis as its length, and its height and width are kept as its original size.

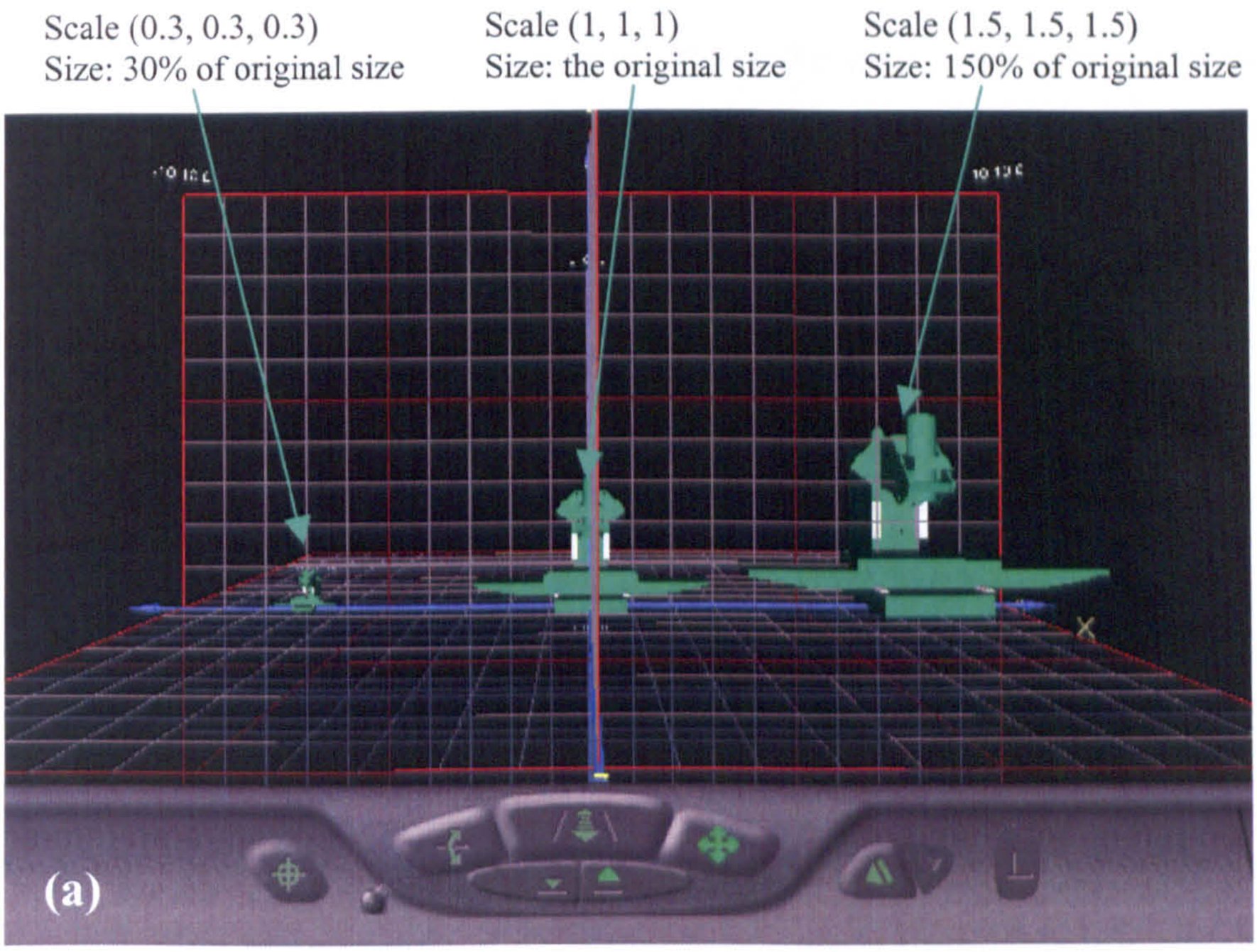


Figure 6-3 The results comparison of different scale input values from the user. (a) Uniform scaling (b) Non-uniform scaling

9. ***Rotation Operation:*** This function allows the user to choose the rotation axis and input the rotation angle for the selected object. An example as shown in Figure 6-4 illustrates the results by rotating the same machine model with different angles around the Y-axis. Of course, the function supports a 3D object to rotate around the X-axis and the Z-axis as well. However, this situation is unlikely happen when manipulating machines in factory layout application in reality. The result is implemented by the procedure:

void __fastcall TForm1::Edit_AngleChange(TObject *Sender)

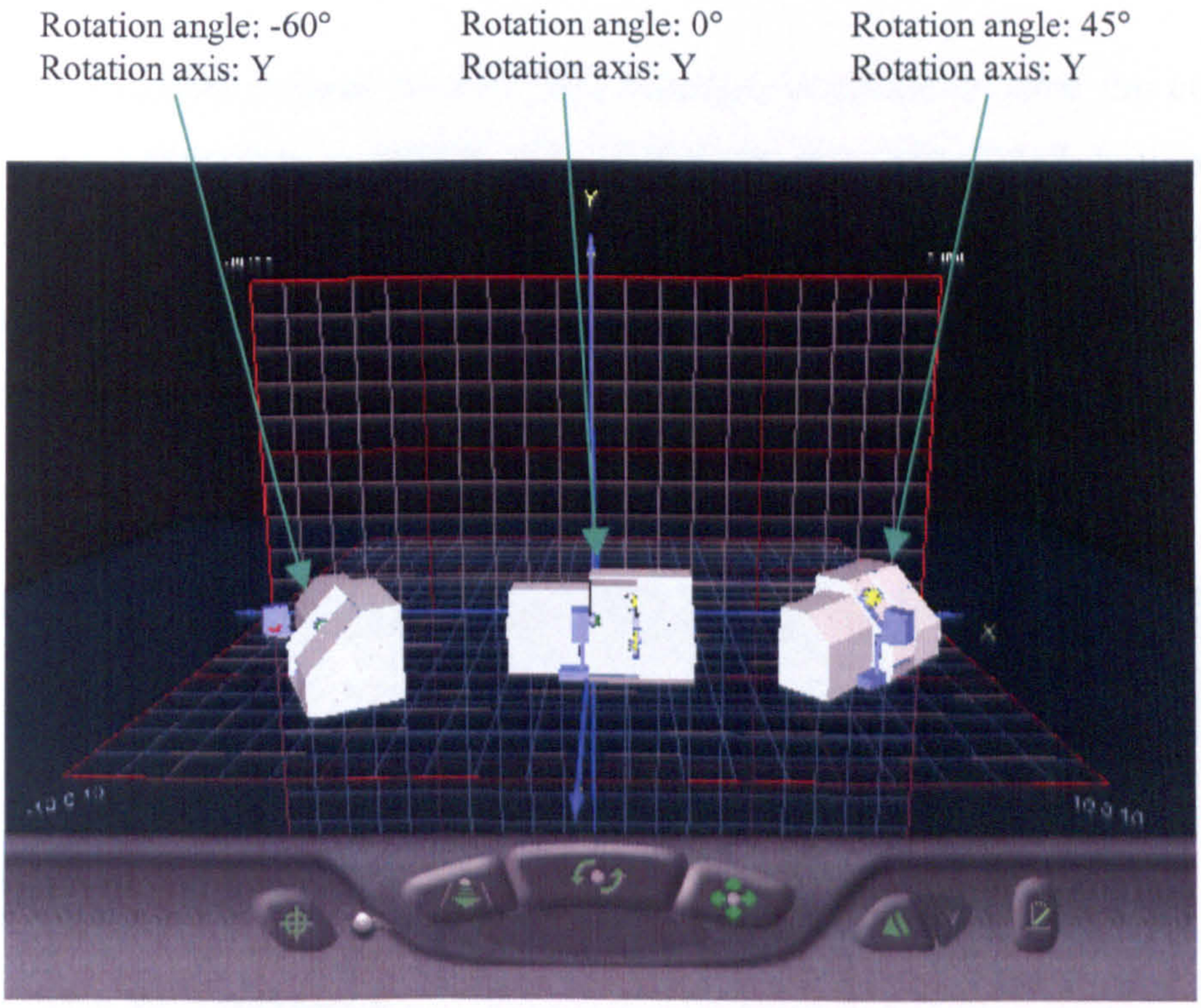


Figure 6-4 The result comparison of different rotation input values from the user.

10. ***Show Current Object Specification:*** This function extracts the relevant engineering information (e.g. machine's specification) from the database.

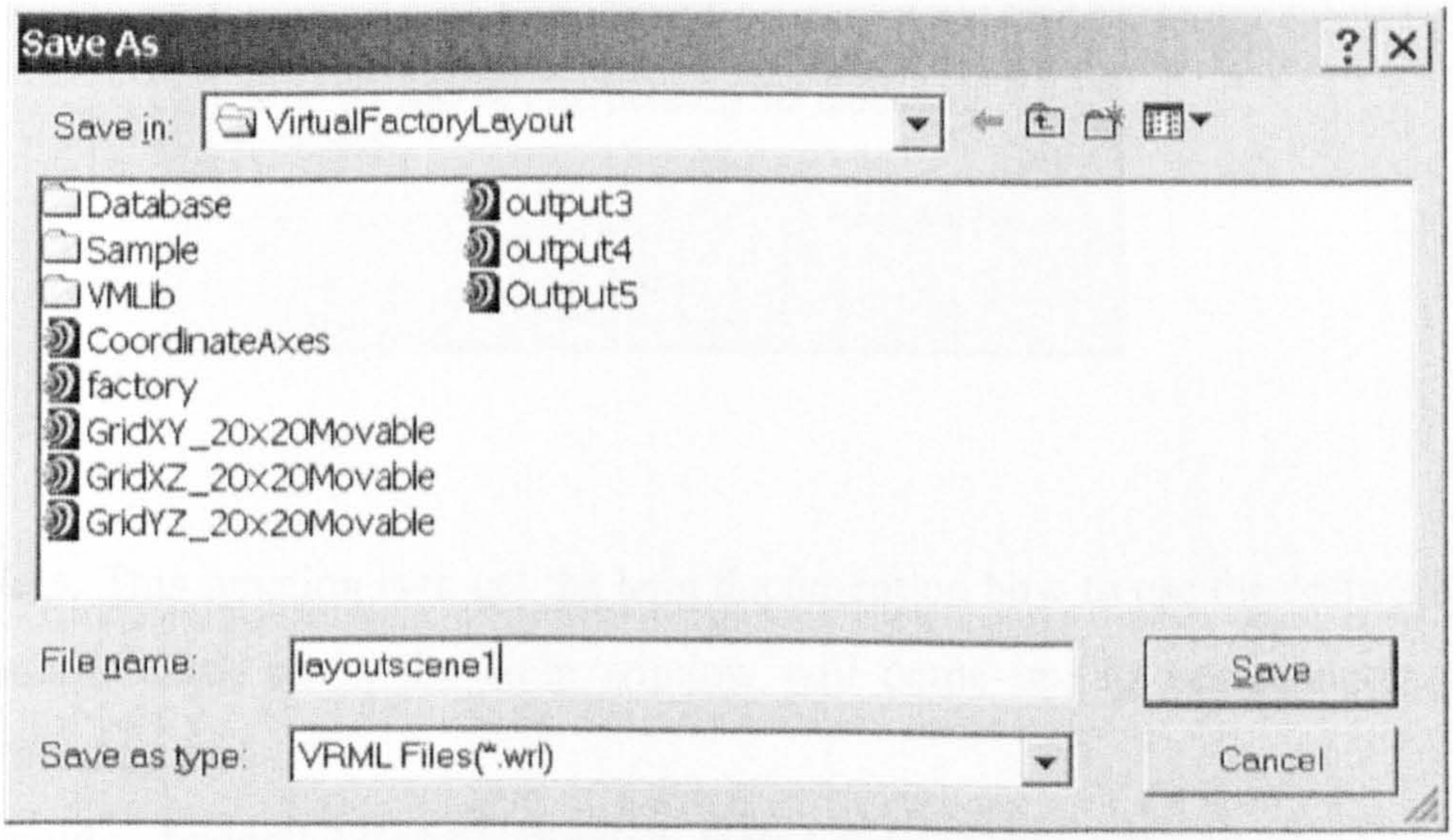
11. ***Embed Web-based Virtual Environment:*** This function allows a Web browser with the CosmoPlayer plug-in to be embedded to provide the Web-based virtual environment.

12. ***Add Object (Integrate geometric data):*** This function allows adding the selected object into the VRML-based scene displayed in Web browser embedded in 11. The result is implemented by the procedure:

    void __fastcall TForm1::Button_AddClick(TObject *Sender)

13. ***Save Current Layout Scene:*** This function executes to save the current layout scene into a VRML file. Click on the "Save As" button, the following dialog comes up allows the user to save a filename in VRML2.0 for the current layout scene. Its corresponding procedure in the source code is:

    void __fastcall TForm1::Button_SaveClick(TObject *Sender)



14. ***Clear Current Virtual Scene:*** This function is to clear the current layout scene and reset the original 3D Grid-enhanced coordinate system in VRML browser shown in 11. The result is implemented by the procedure:

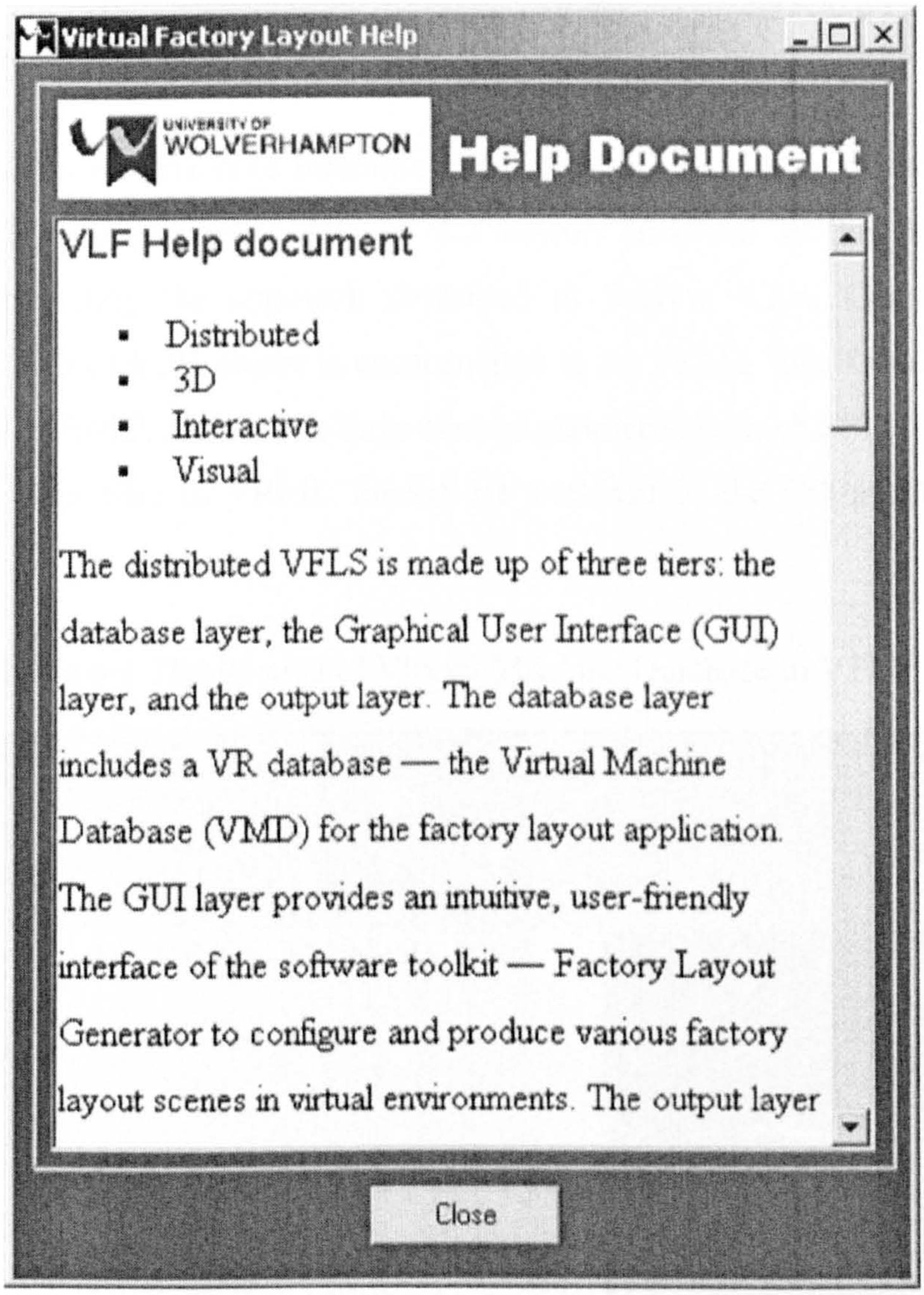
    void __fastcall TForm1::Button_ClearClick(TObject *Sender)

15. ***About:*** This function is to get the information on the software package. Click on it, the following window will come up. Its implementation procedure is:

    void __fastcall TForm1::Button_AboutClick(TObject *Sender)

16. ***Help:*** This function is to get the help document on how to use the software toolkit. Click on it, the Help window will come up. Its corresponding procedure is:

    void __fastcall TForm1::Button_HelpClick(TObject *Sender)



17. ***Reset:*** This function is to reset all input values from the user to system default: position (0, 0, 0) for move group, the original size of object (1, 1, 1) for the scale group, and Y-axis for rotation axis and the initial rotation angle is 0° for rotation group. The result is implemented by the following procedure:

    void __fastcall TForm1::Button_ResetClick(TObject *Sender)

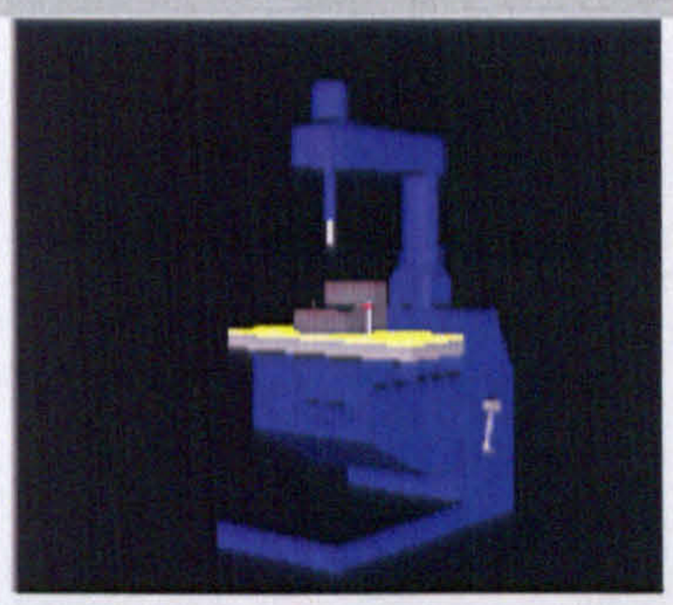
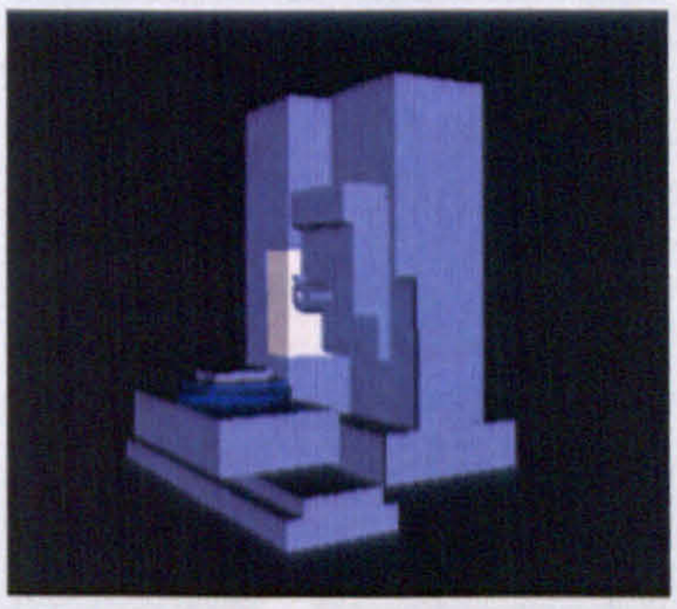
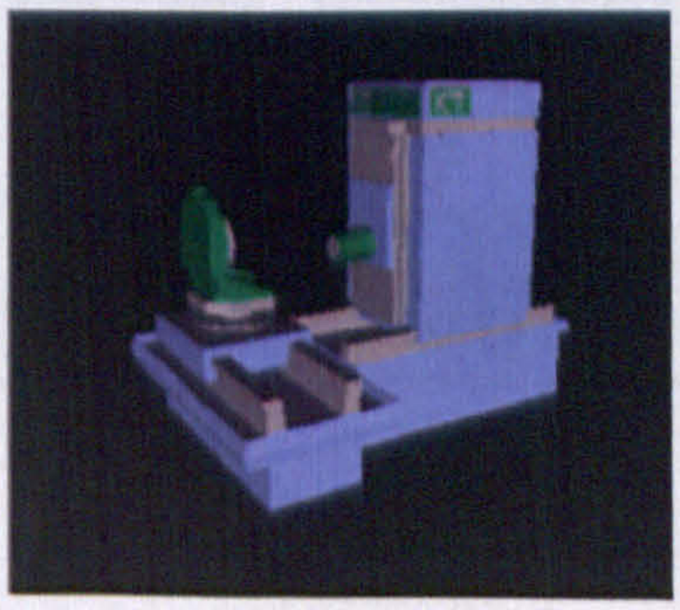
18. ***Quit:*** This function allows the user to quit the software toolkit. Click on it, the program will exit. Its corresponding procedure is:

    void __fastcall TForm1::Button_QuitClick(TObject *Sender)

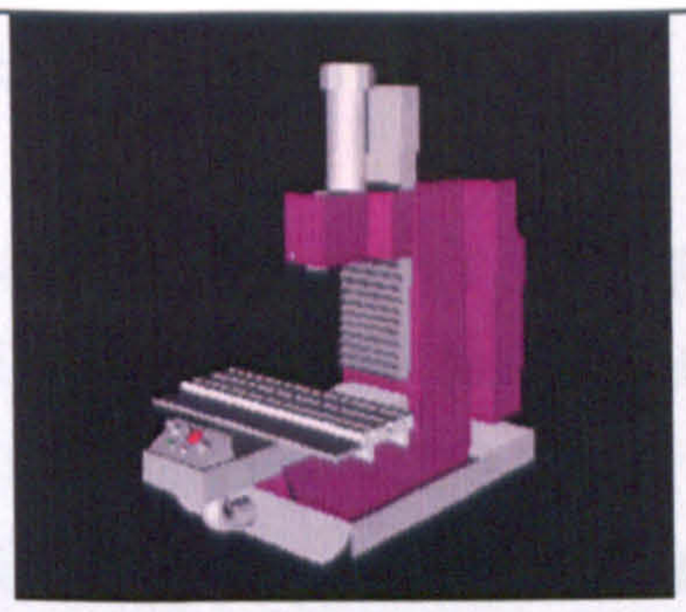
6.2.1 Virtual Machine Database

At present, Virtual Machine Database (VMD) in VFLS consists of the following 3D objects. The geometric data of 3D objects involved in the database are obtained by using the approach described in Section 4.2.4, Chapter 4. The geometric data of a 3D object is encapsulated in its VRML file. Table 6-1 shows the results of VMD. Due to the large size of geometric data of a 3D object, these 3D object data files in VRML format are enclosed in the CD attached to this thesis.

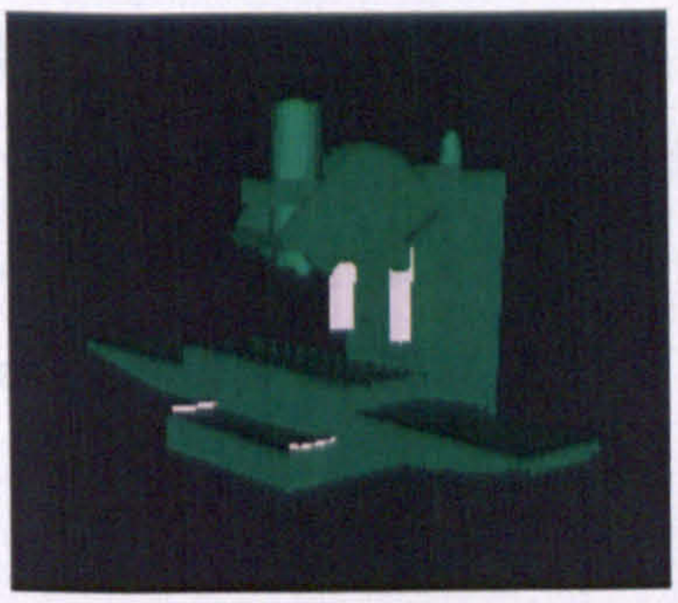
Table 6-1 The Results of Virtual Machine Database in VFLS

Category	Name	VRML file (Geometric data)	3D View
Category1	Model1-1	Machine1-1.wrl	
Category1	Model1-2	Machine1-2.wrl	
Category1	Model1-3	Machine1-3.wrl	

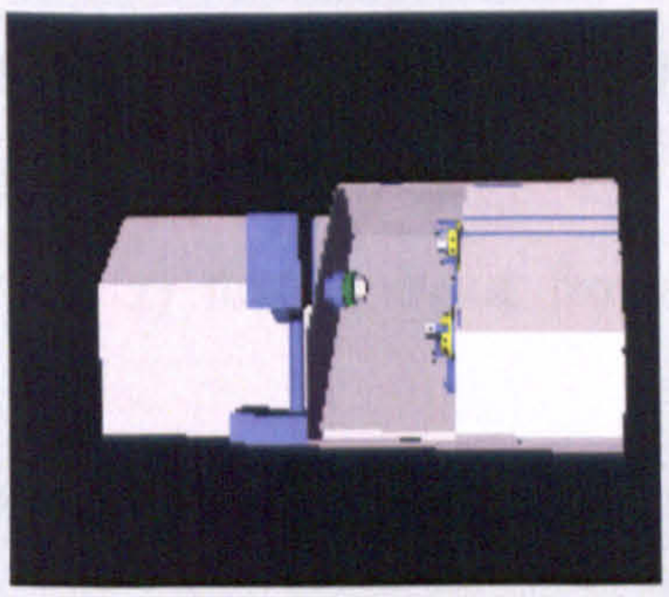
Category2 Model2-1 Machinel2-1.wrl



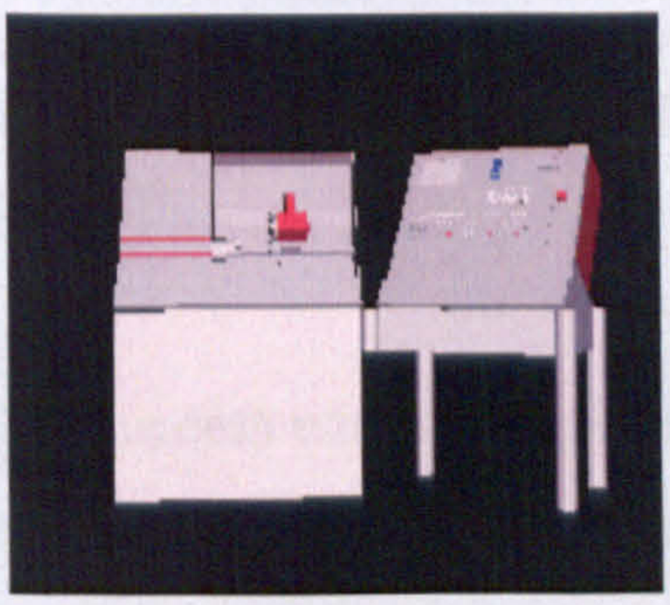
Category2 Model2-2 Machine2-2.wrl



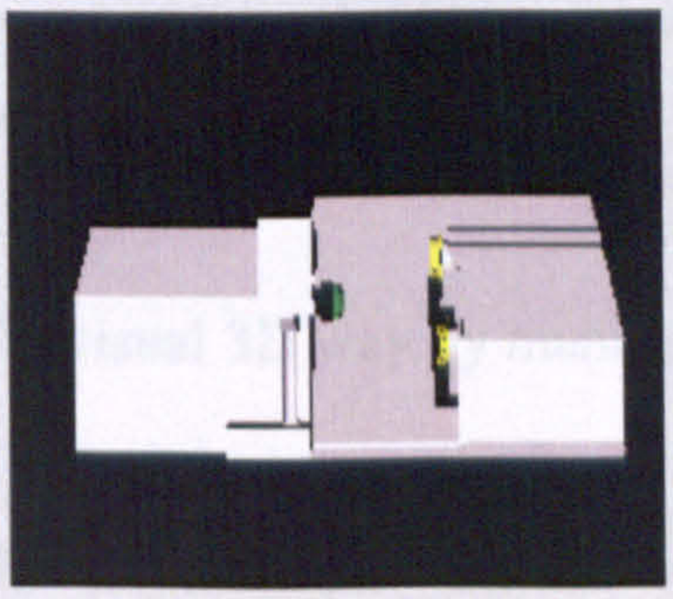
Category3 Model3-1 Machine3-1.wrl



Category3 Model3-2 Machinel3-2.wrl



Category3 Model3-3 Machine3-3.wrl



6.2.2 VRML-based factory layout experimental results

Figure 6-5 through Figure 6-10 show some experimental results as examples of general factory layout scenes which are used to validate the demonstrator — VRML-based Factory Layout Simulator which was developed on the basis of the Web-based VR approach for design and manufacturing proposed by the author in this project. These factory layout scenes have been successfully produced in the VFLS via direct interactive manipulation. The geometric data of each model are selected from the Virtual Machine Database (VMD) and the initial input values from users are shown in corresponding tables along with the layout scene. There are:

- ***Category:*** an object's category defined in the Virtual Machine Database.
- ***Model:*** an object's filename that contains its geometric data.
- ***Position:*** an object's position value (x, y, z) that is input from move operation.
- ***Scale:*** an object's size and its scale value (x, y, z) that is input in scale operation.
- ***Orientation:*** an object's orientation and its rotation angle (rotation axis and angle) that is input in rotation operation.

These experimental results show that the VFLS has success with the generation of layout scenes according to the input values from the user. And the ***virtual sensor*** bound on each object (machine) provides the enhancement of the interactive layout operation. The software toolkit has been implemented to

- Quickly produce various factory layouts in visual 3D way by manipulating 3D models of machines.
- Deliver 3D visual scenarios of the factory layouts across the Internet.
- Allow distributed Web-based users to interact with the virtual 3D factory layout environments
- Be compatible with Web-based VR standard —VRML.

Category	Model	Position (x, y, z)	Size (x, y, z)	Orientation (Angle & axis)
Category1	Model1-1	(-5, 0, 7)	(1, 1, 1)	0°
Category1	Model1-1	(-5, 0, 0)	(1, 1, 1)	0°
Category1	Model1-1	(-5, 0, -7)	(1, 1, 1)	0°
Category3	Model3-3	(5, 0, 7)	(1, 1, 1)	45°, Y-axis
Category3	Model3-1	(5, 0, 0)	(1, 1, 1)	45°, Y-axis
Category3	Model3-2	(5, 0, -7)	(1, 1, 1)	45°, Y-axis

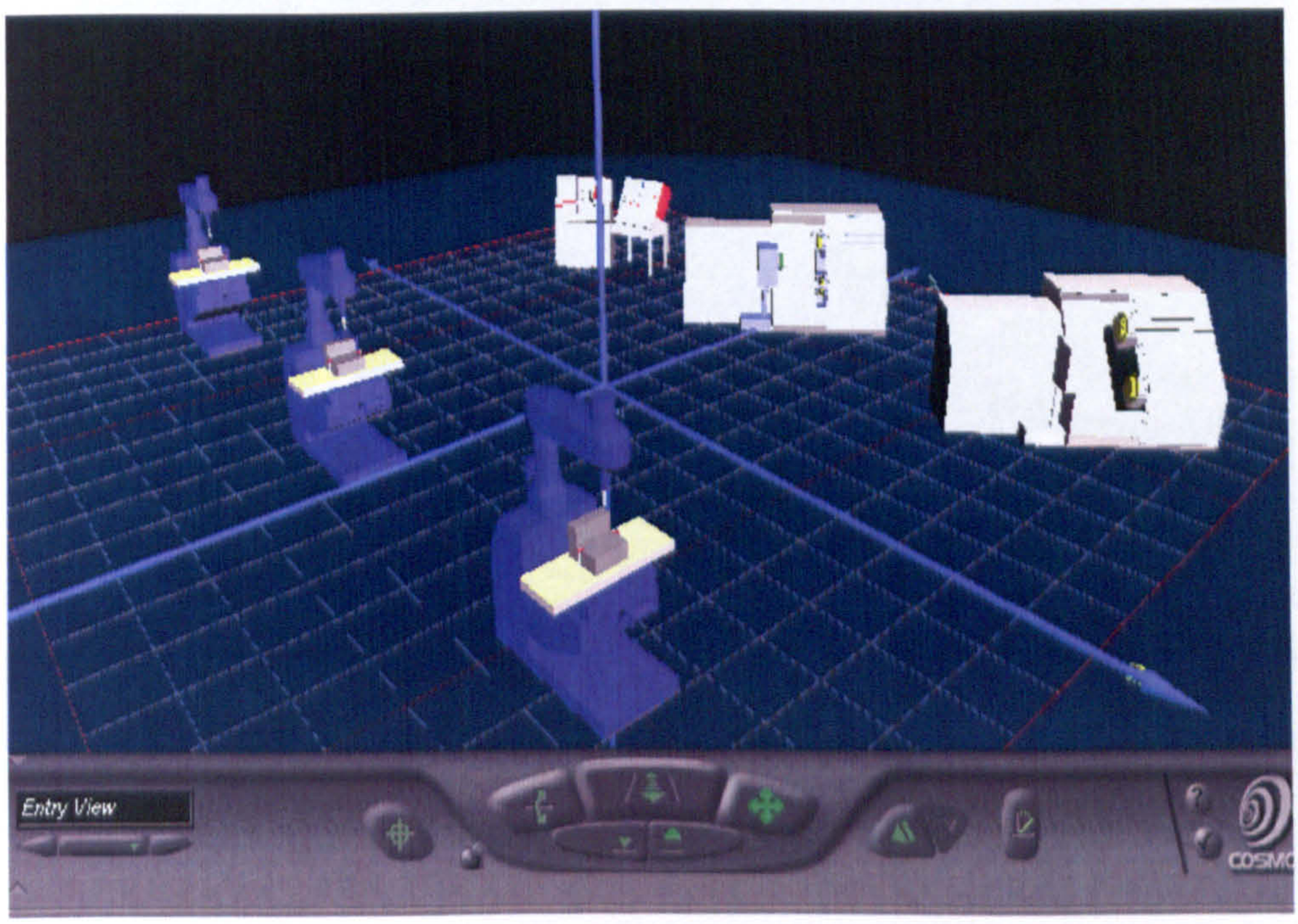


Figure 6-5 Experimental Result 1 — the result of a VRML-based factory layout scene produced in VFLS by the user input values.

Category	Model	Position (x, y, z)	Size (x, y, z)	Orientation (Angle & axis)
Category2	Model2-1	(-7, 0, 0)	(1.2, 1.2, 1.2)	90°, Y-axis
Category1	Model1-3	(-7, 0, -7)	(1.2, 1.2, 1.2)	0°
Category1	Model1-2	(0, 0, -7)	(1.2, 1.2, 1.2)	0°
Category3	Model3-2	(7, 0, -7)	(1.2, 1.2, 1.2)	0°
Category3	Model3-3	(8, 0, 7)	(1.2, 1.2, 1.2)	-90°, Y-axis

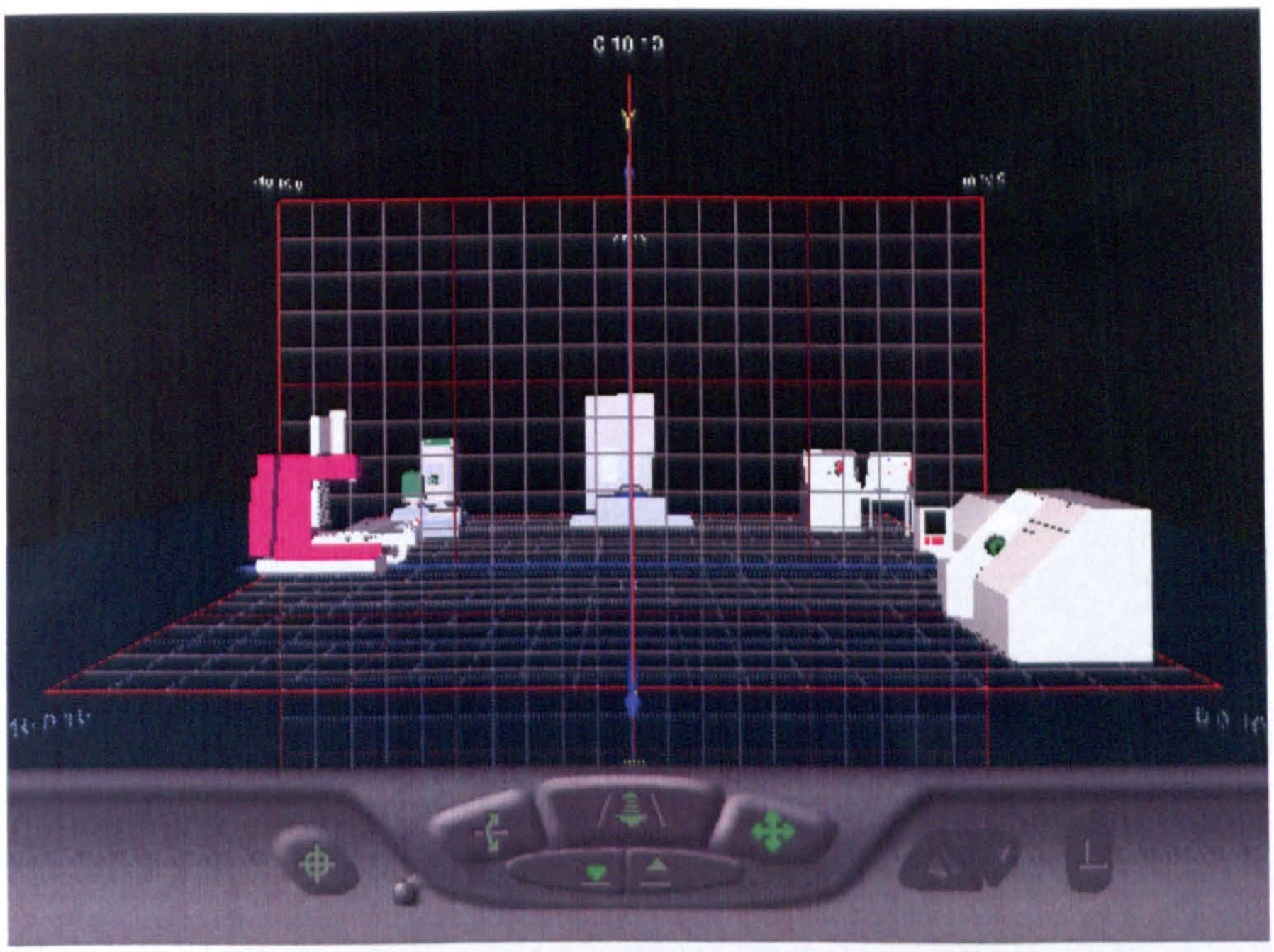


Figure 6-6 Experimental Result 2 — the result of a VRML-based factory layout scene produced in VFLS and its input values from the user.

Category	Model	Position (x, y, z)	Size (x, y, z)	Orientation (Angle & axis)
Category1	Model1-1	(-7, 0, 7)	(1.5, 1.5, 1.5)	90°, Y-axis
Category1	Model1-1	(-7, 0, 0)	(1.5, 1.5, 1.5)	90°, Y-axis
Category1	Model1-1	(-7, 0, -7)	(1.5, 1.5, 1.5)	90°, Y-axis
Category2	Model2-2	(0, 0, 4)	(1.5, 1.5, 1.5)	0°
Category2	Model2-2	(0, 0, -7)	(1.5, 1.5, 1.5)	0°
Category2	Model2-1	(7, 0, -7)	(1.5, 1.5, 1.5)	0°
Category1	Model1-2	(7.5, 0, 7)	(1.5, 1.5, 1.5)	-90°, Y-axis

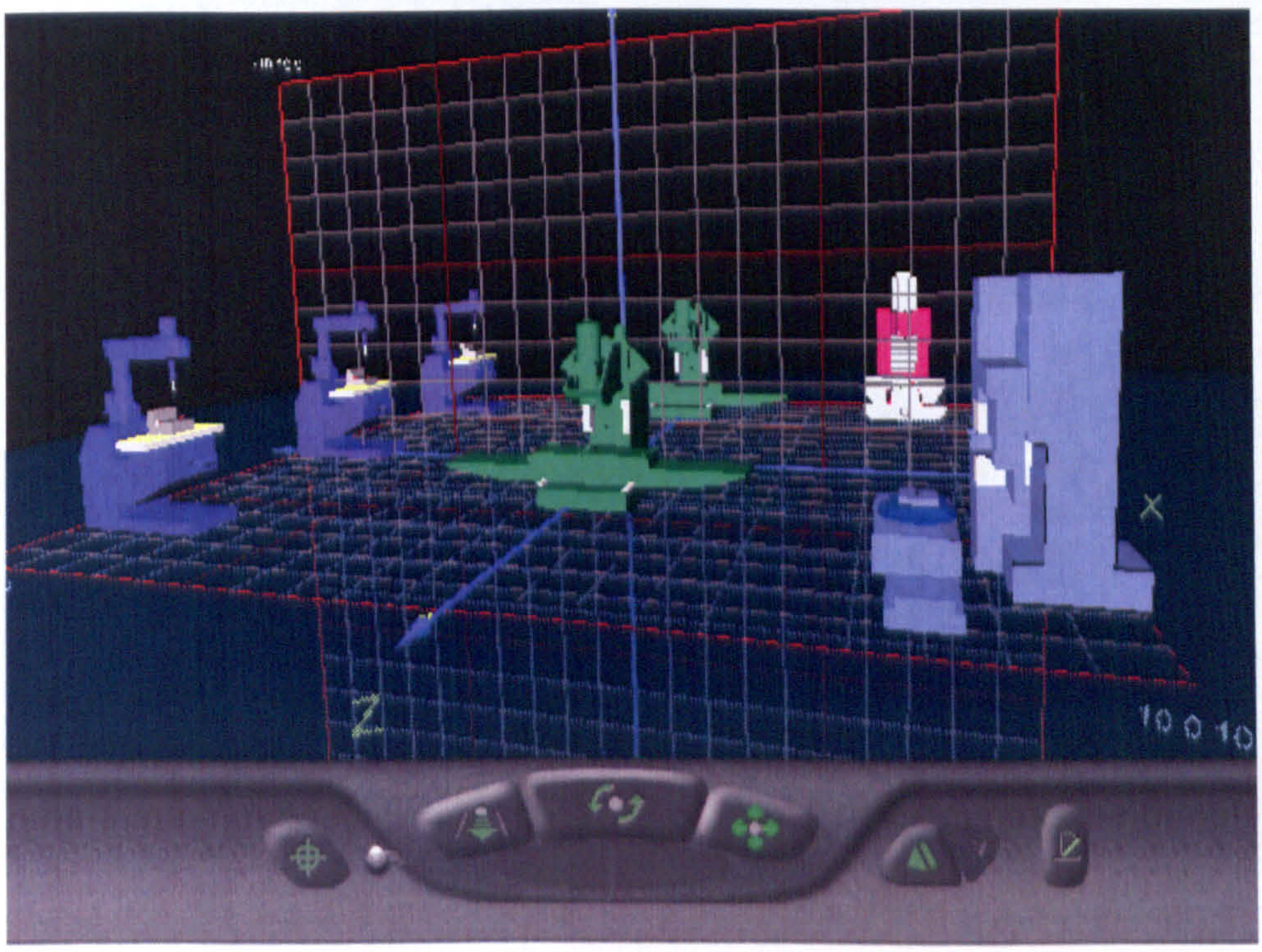


Figure 6-7 Experimental Result 3 — the result of a VRML-based factory layout scene produced in VFLS and its input values from the user.

Category	Model	Position (x, y, z)	Size (x, y, z)	Orientation (Angle & axis)
Category3	Model3-1	(-6.5, 0, 0)	(1.3, 1.3, 1.3)	90°, Y-axis
Category3	Model3-2	(-5, 0, -6.5)	(1.3, 1.3, 1.3)	90°, Y-axis
Category2	Model2-1	(7, 0, -7)	(1.3, 1.3, 1.3)	0°
Category2	Model2-1	(7, 0, 0)	(1.3, 1.3, 1.3)	0°
Category2	Model2-1	(7, 0, 7)	(1.3, 1.3, 1.3)	0°

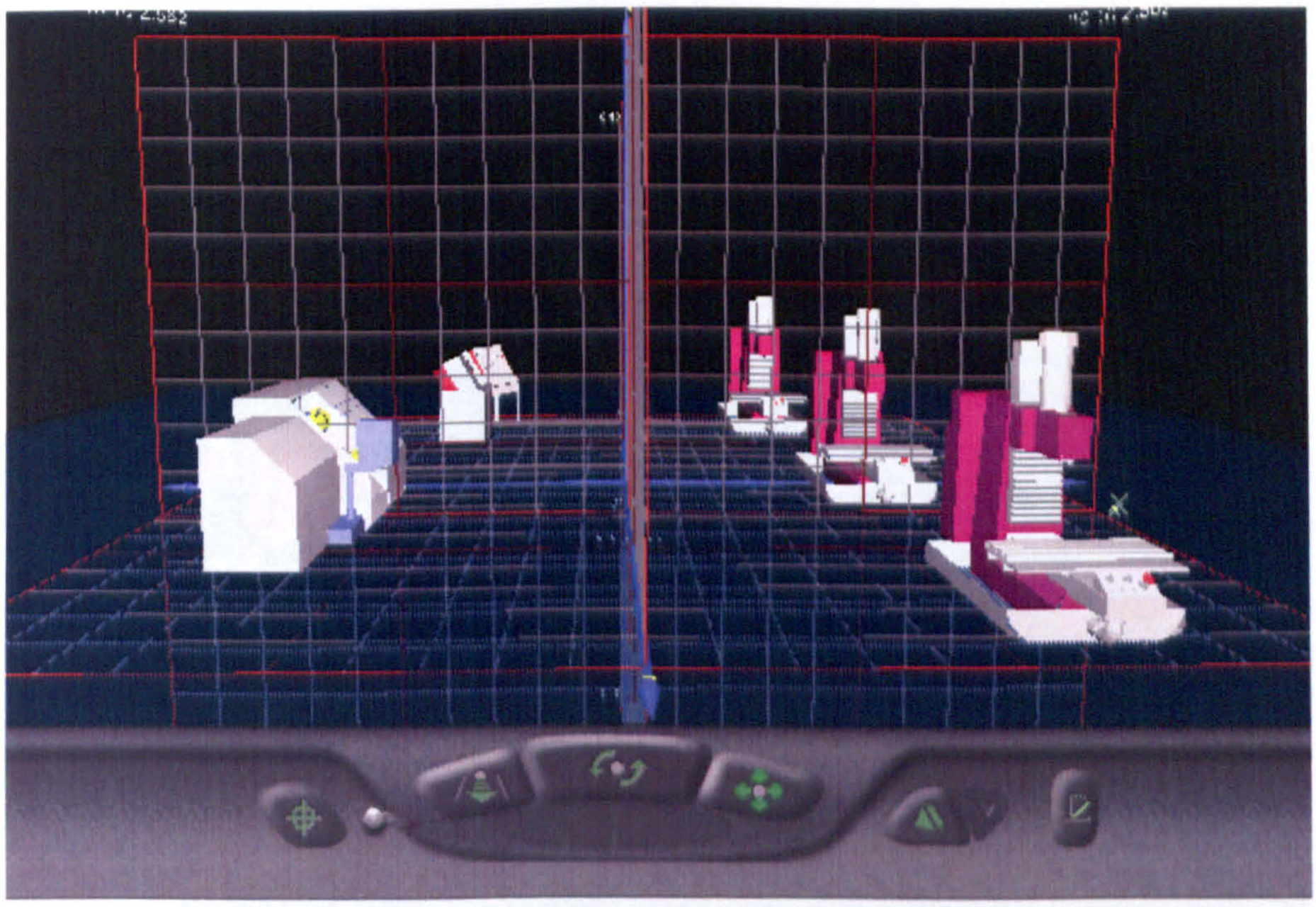


Figure 6-8 Experimental Result 4 — the result of a VRML-based factory layout scene produced in VFLS and its input values from the user.

Category	Model	Position (x, y, z)	Size (x, y, z)	Orientation (Angle & axis)
Category2	Model2-1	(-7.5, 0, 7.5)	(1.2, 1.2, 1.2)	90°, Y-axis
Category2	Model2-1	(-7.5, 0, 0)	(1.2, 1.2, 1.2)	90°, Y-axis
Category2	Model2-1	(-7.5, 0, -7.5)	(1.2, 1.2, 1.2)	90°, Y-axis
Category1	Model1-3	(6.5, 0, -7.5)	(1.2, 1.2, 1.2)	-90°, Y-axis
Category1	Model1-3	(6.5, 0, 0)	(1.2, 1.2, 1.2)	-90°, Y-axis
Category1	Model1-3	(6.5, 0, -7.5)	(1.2, 1.2, 1.2)	-90°, Y-axis

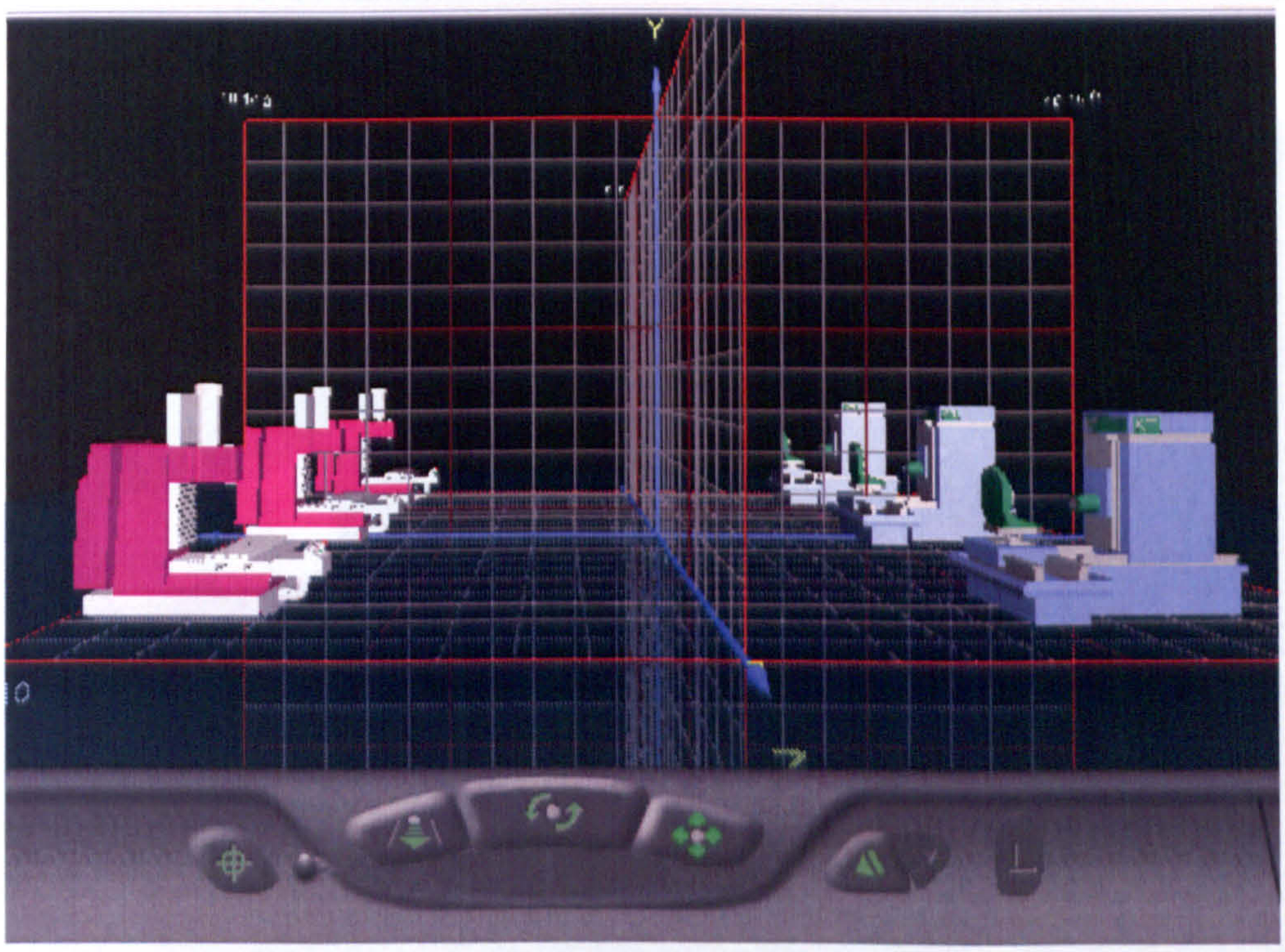


Figure 6-9 Experimental Result 5 — the result of a VRML-based factory layout scene produced in VFLS and its input values from the user.

Category	Model	Position (x, y, z)	Size (x, y, z)	Orientation (Angle & axis)
Category3	Model3-1	(-7, 0, 7)	(1, 1, 1)	45°, Y-axis
Category3	Model3-1	(0, 0, 7)	(1, 1, 1)	45°, Y-axis
Category3	Model3-1	(7, 0, 7)	(1, 1, 1)	45°, Y-axis
Category1	Model1-1	(-7, 0, 0)	(1, 1, 1)	90°, Y-axis
Category1	Model1-1	(0, 0, 0)	(1, 1, 1)	90°, Y-axis
Category1	Model1-1	(7, 0, 0)	(1, 1, 1)	90°, Y-axis
Category2	Model2-1	(-7.5, 0, -7)	(1, 1, 1)	0°
Category2	Model2-1	(0, 0, -7)	(1, 1, 1)	0°
Category2	Model2-1	(8, 0, -7)	(1, 1, 1)	0°

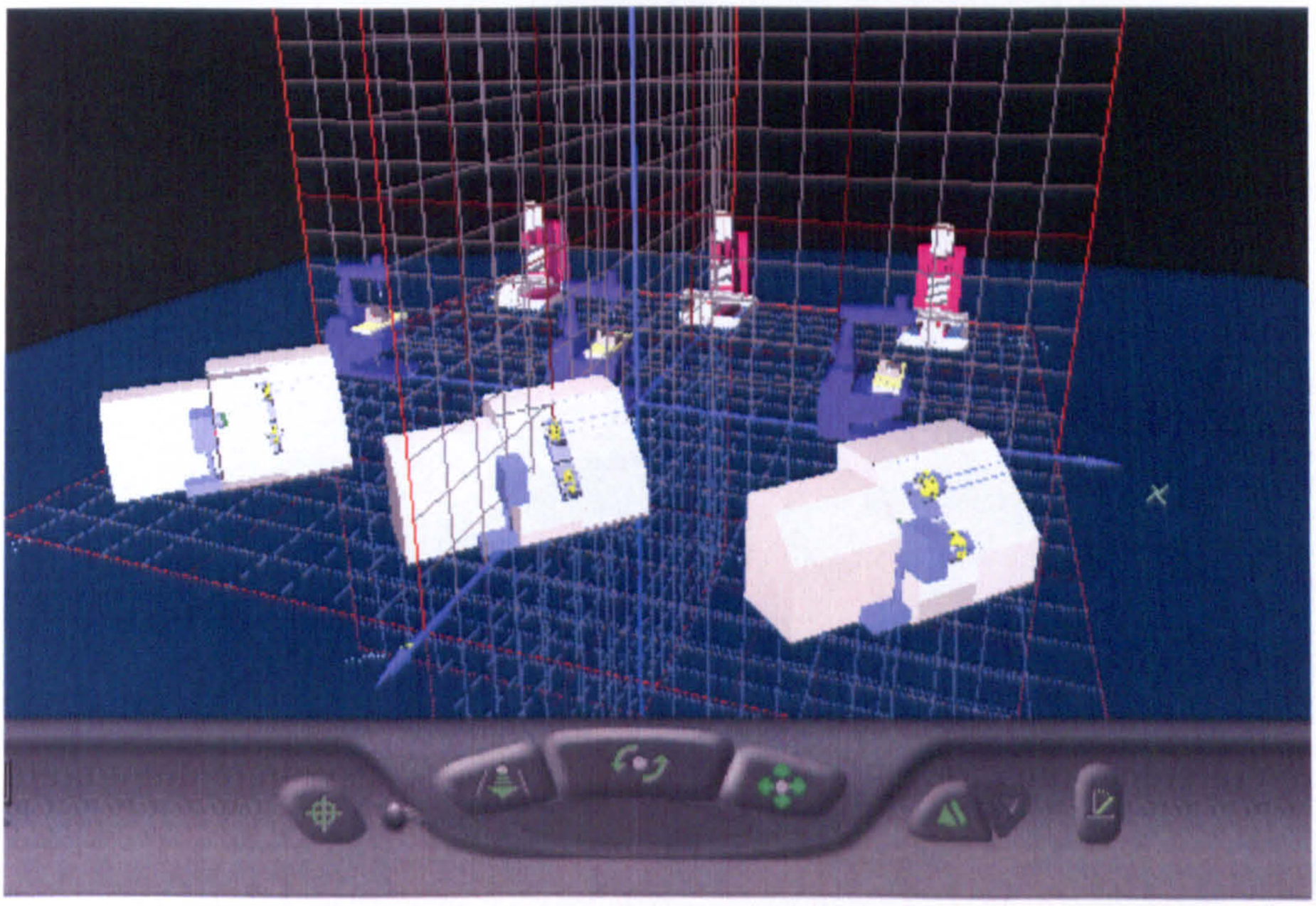


Figure 6-10 Experimental Result 6 — the result of a VRML-based factory layout scene produced in VFLS and its input values from the user.

6.2.3 Multiple-user collaboration working experimental result

The Web-based VR system can be incorporated in a networking environment to support collaboration working. This section presents the result of collaboration working experiment among multiple users. The collaboration working experiment has successfully validated that the Web-based VR software system developed by the author can support engineering activities (e.g. factory layout) of design and manufacturing in the collaboration network environment.

This experiment aims to enable multiple users at different locations to analyze, review, and interact with the 3D real-time interactive simulations presented in the VFLS for collaboration working over a local or wide area network. Firstly, three common computers were connected in this experiment by the author via an Ethernet switch to set up a local area networking environment as shown in Figure 6-11. The photos of experimental hardware facilities are shown in Figure 6-12 and their specifications are listed in Table 6-2.

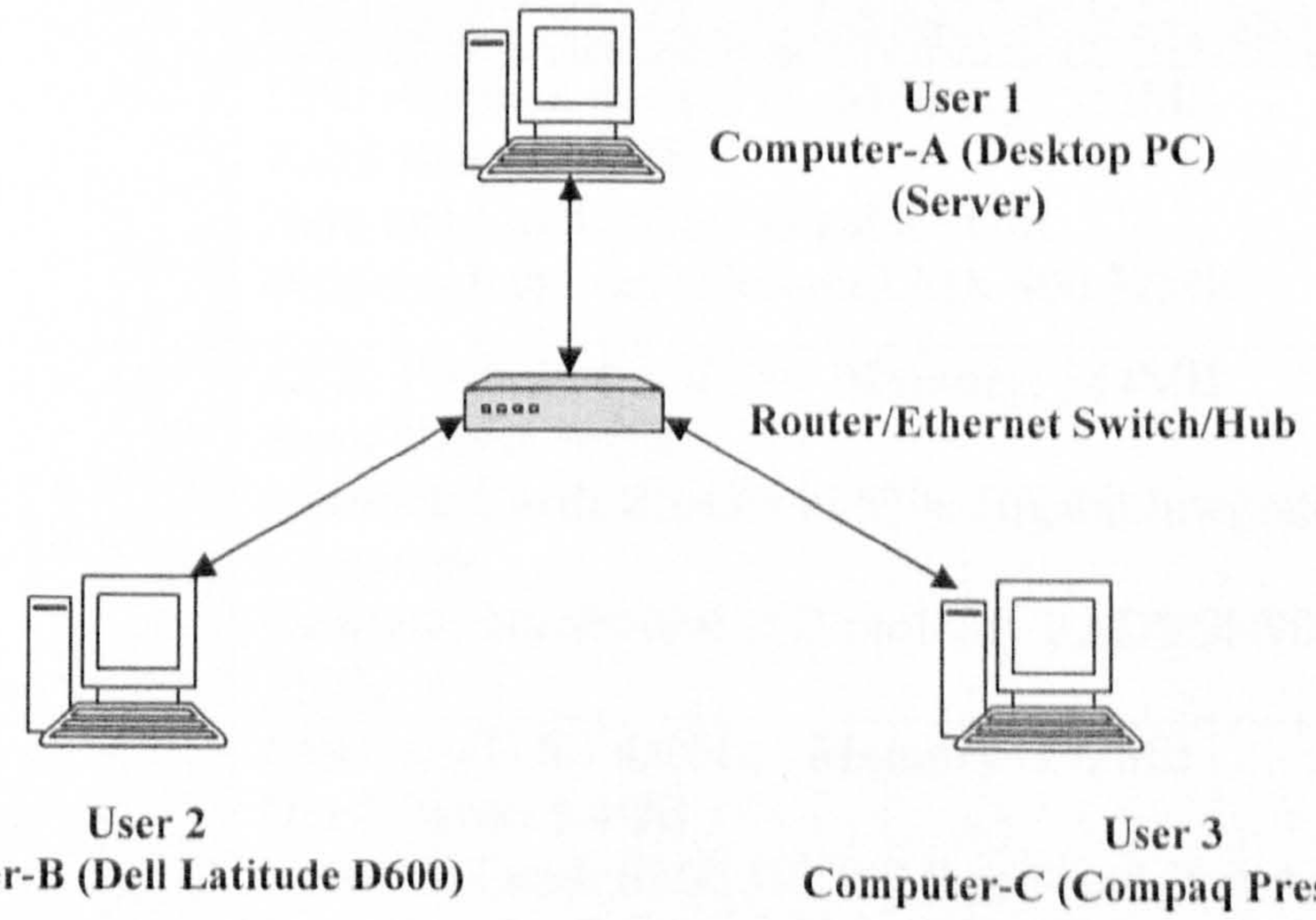


Figure 6-11 Multiple-user collaboration working experiment – three computers were connected via an Ethernet switch to set up a local area networking environment

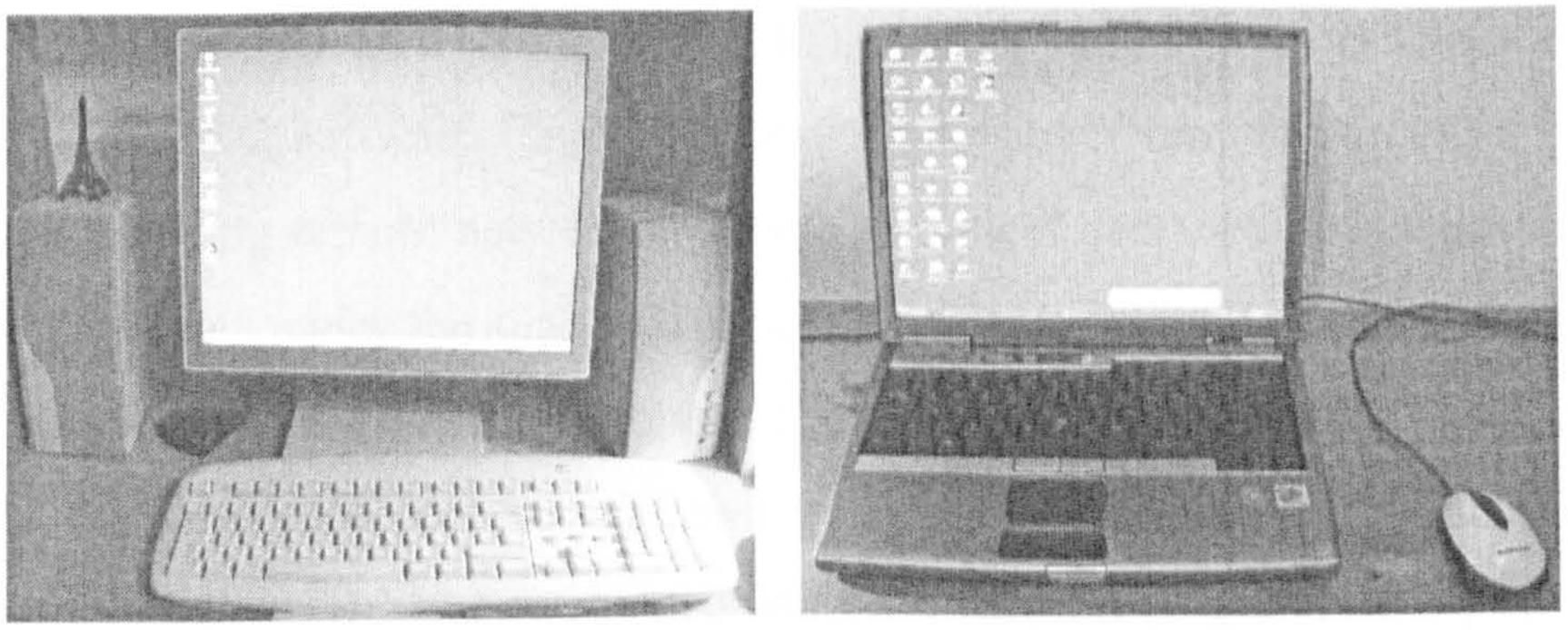


Figure 6-12 Multiple-user collaboration working experiment – the photos of experimental hardware facilities

Hardware Resource	Specifications
Computer-A **Desktop PC**	**CPU:** Pentium 4 2.8GHz **Memory**: 512MB **Hard Drive:** 80GB **Network Card:** 3Com Gigabit LOM Graphics Hardware: Geforce2 MX 400 32MB
Computer-B **Dell Latitude D600**	**CPU:** Pentium 4 2.0GHz **Memory**: 384MB **Hard Drive:** 30GB **Network Card:** Broadcom 570x Gigabit Integrated Controller **Graphics Hardware:** ATI mobility RADEON9000 32MB
Computer-C **Compaq Presario 1690**	**CPU:** AMD K3 450Hz **Memory**: 192MB **Hard Drive:** 6.4GB **Network Card:** SMC 10/100Mbps Fixed-Port Adapter **Graphics Hardware:** ATI RAGE LT PRO 8MB
Ethernet Switch **GIGABYTE Mini Ethernet Switch**	**Number of port:** 4 ports **Speed:** 10/100Mbps

Table 6-2 The specifications of hardware facilities in multiple-user collaboration working experiment

This experiment set up a virtual networking collaboration environment with the aid of TightVNC software. TightVNC is a client/server remote control software package allowing remote network access to graphical desktops. It is a useful free toolkit (released under the GNU General Public License) which is available for both Windows and UNIX to support cross-platform collaboration working in remote administration, remote customer support, education, and for many other purposes. It can be freely downloaded from the Internet (TightVNC 2003). With TightVNC, user can see the desktop of a remote machine and control it with local mouse and keyboard, just like the user would do it sitting in the front of that remote computer. The mouse cursor movements do not generate screen updates anymore, remote cursor movements are processed locally by the viewer, so users do not see slow remote cursor movements behind the local cursor.

This collaboration working experiment are described as follow and was recorded as a video file which was included in the attached CD:

- TightVNC was respectively installed on the above three computers. The computer-A (Desktop PC) was set up as a server controlled by *User1*, which was installed and run the Web-based VR system — VRML-based Factory Layout Simulator (VFLS) developed by the author in this research project. Other two computers are controlled by *User2* and *User3*.

- *User1* launched ***TightVNC Server*** program on the **Computer-A** and then run the Web-based VR system — VFLS as shown in Figure 6-13.

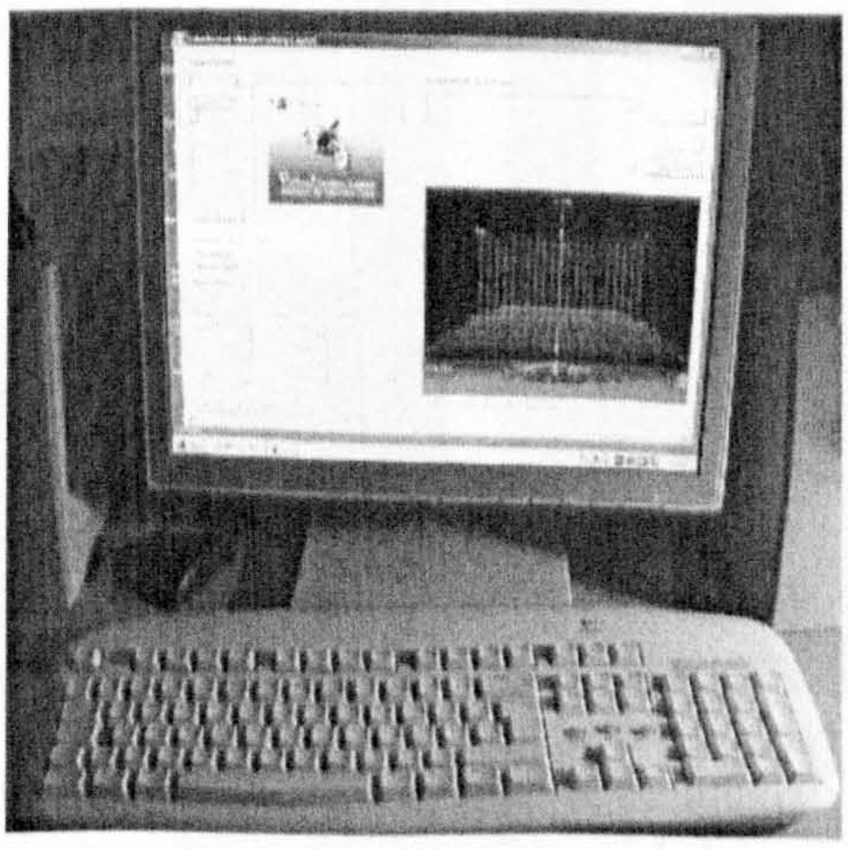


Figure 6-13 The VFLS was installed and run on Computer-A (server) by *User1*

- *User2* and *User3* run ***TightVNC Viewer*** program on their machines (**Computer-B** and **Computer-C**) and then set them in the ***Listen Mode***. Then, *User2* and *User3* were requested to logon the remote server (**Computer-A**) with username and password as shown in Figure 6-14.

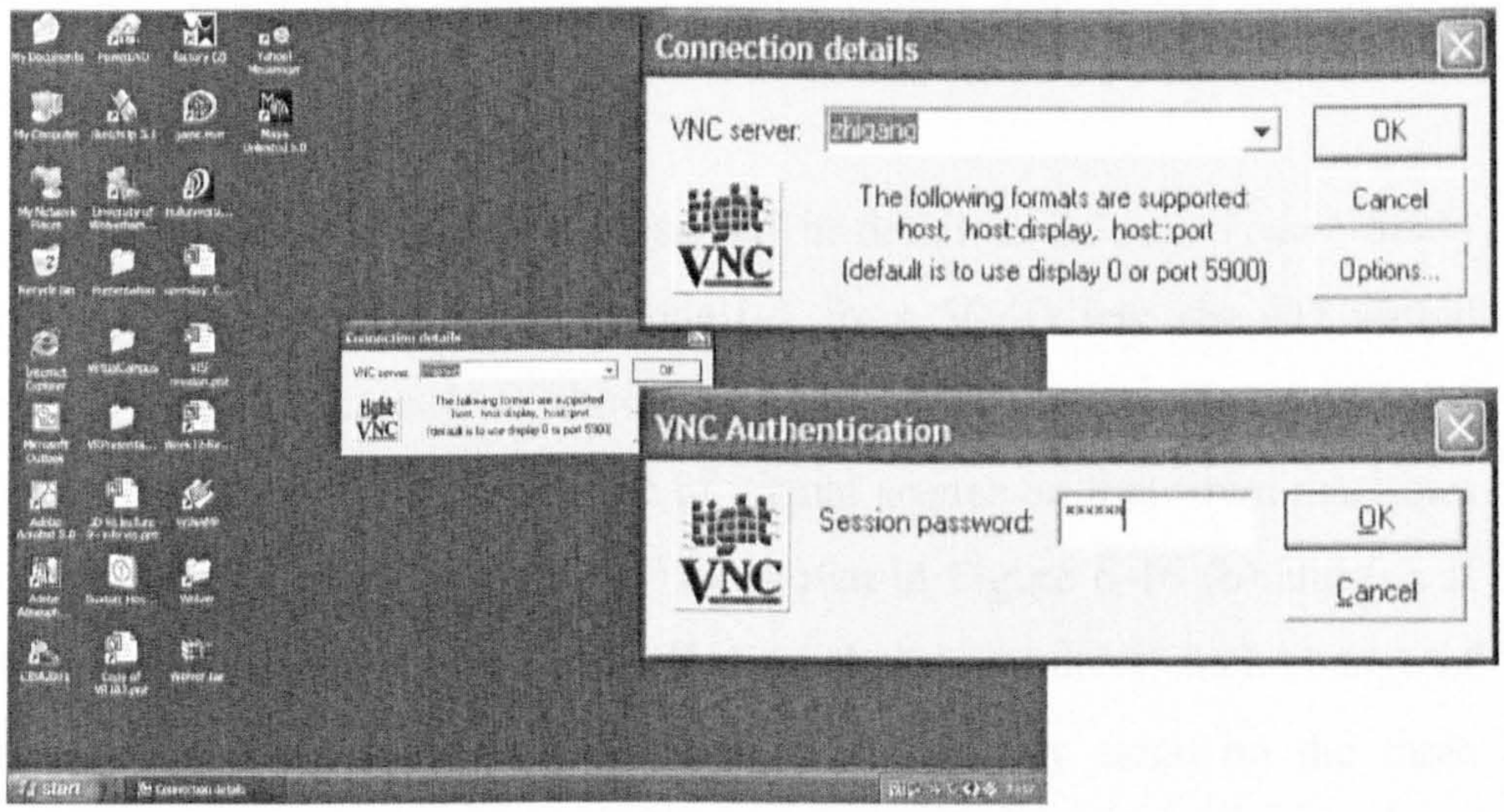


Figure 6-14 *User2* and *User3* logon the remote server (Computer-A) via username and password

- Once logon the server (**Computer-A**), *User2* and *User3* saw the desktop of **Computer-A** and controlled it with local mouse and keyboard, just like the user would do it sitting in the front of **Computer-A**. Thus, *User2* and *User3* can access and interact with the VFLS which was only installed on the remote machine (**Computer-A)** as shown in Figure 6-15.

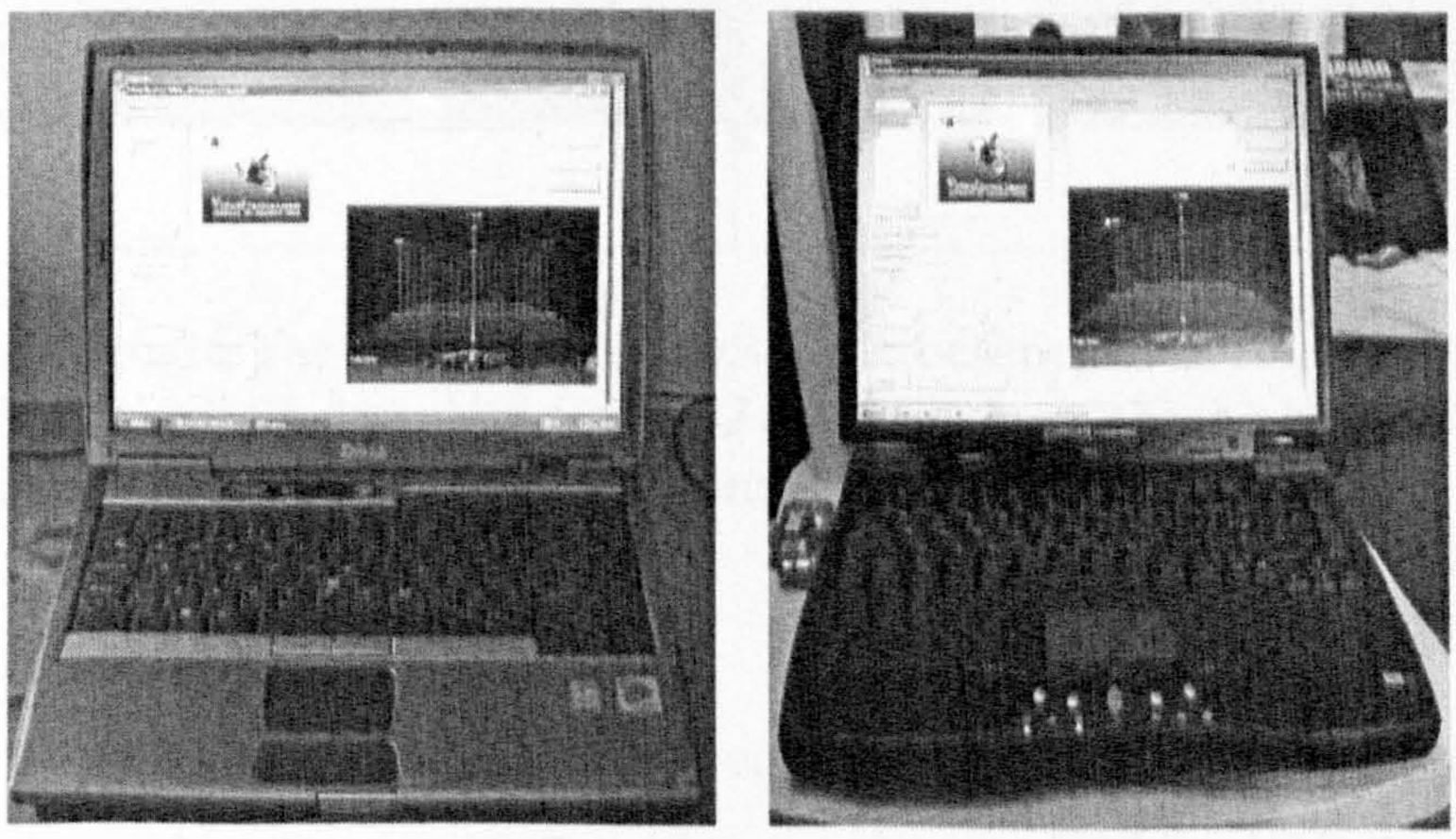


Figure 6-15 *User2* and *User3* can access and interact with the VFLS (run on the Computer-A) and control it with local mouse and keyboard

- The multiple users (e.g. *User1*, *User2*, and *User3*) controlled the Web-based VR system — VFLS to produce the factory layout scenes via collaboration working in the networking environment.

- The communications among users were implemented using typing-text in Notepad program to discuss and decide which user would like to take control.

- An experimental example is presented in detail as below. *User1* firstly controlled the VFLS to add Model1-1 from VMD into the 3D virtual world on **Computer-A** as shown in Figure 6-16 (a). At the same time, *User2* and *User3* saw the change of virtual scenes on their own machines (**Computer-B** and **Computer-C**) as shown in Figure 6-16 (b) and (c). If *User2* or *User3* moved the Model1-1 on their computers, such change of the virtual scenes will be presented as the exactly same on the three computers.

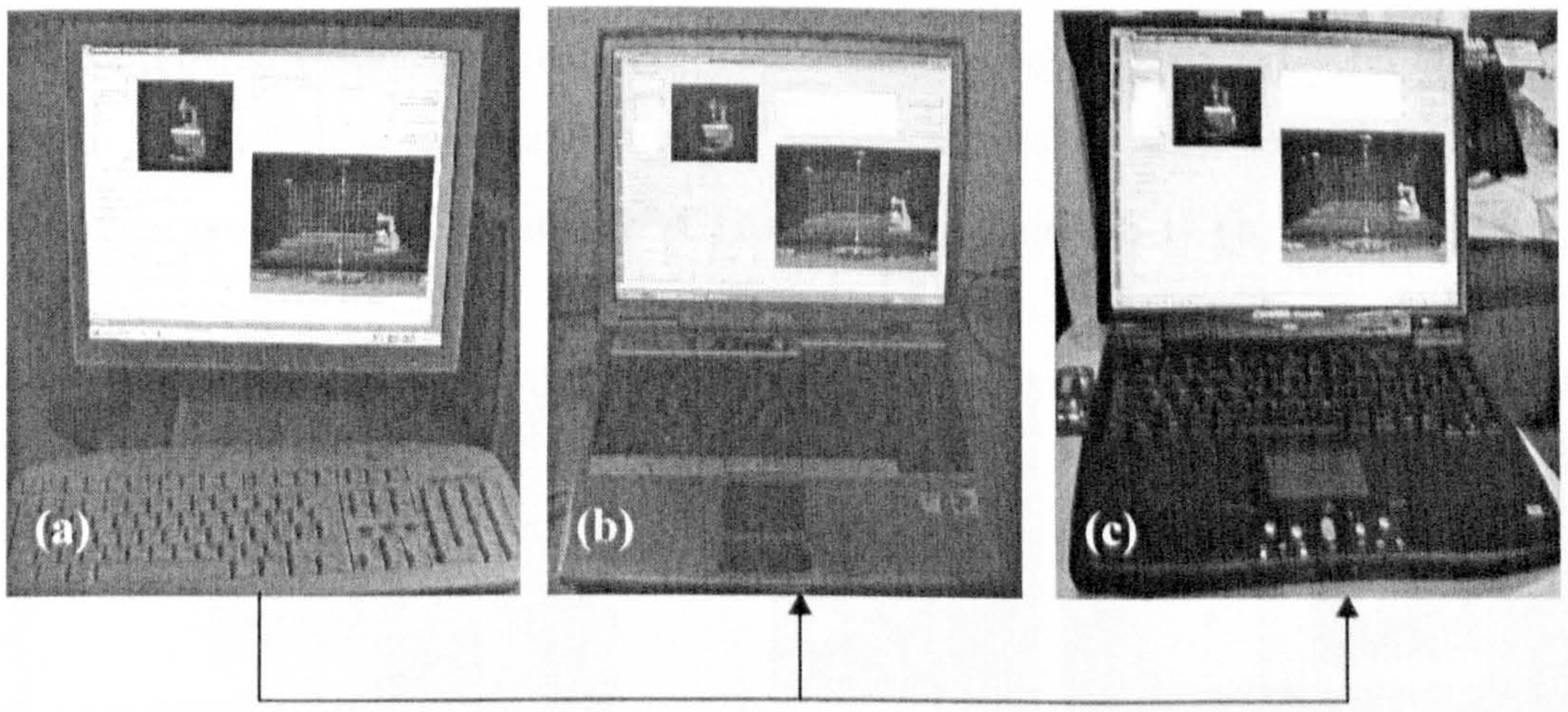


Figure 6-16 The collaboration working experimental example 1 – *User1* controlled the VFLS, *User2 and User3* can see the change of virtual scenes on their own machines

User2 can also control the VFLS to add Model2-1 from VMD into the 3D virtual world on **Computer-B** as shown in Figure 6-17 (a). And *User1* and

User3 then saw the change of virtual scenes on their own machines (**Computer-A** and **Computer-C**) as shown in Figure 6-17 (b) and (c).

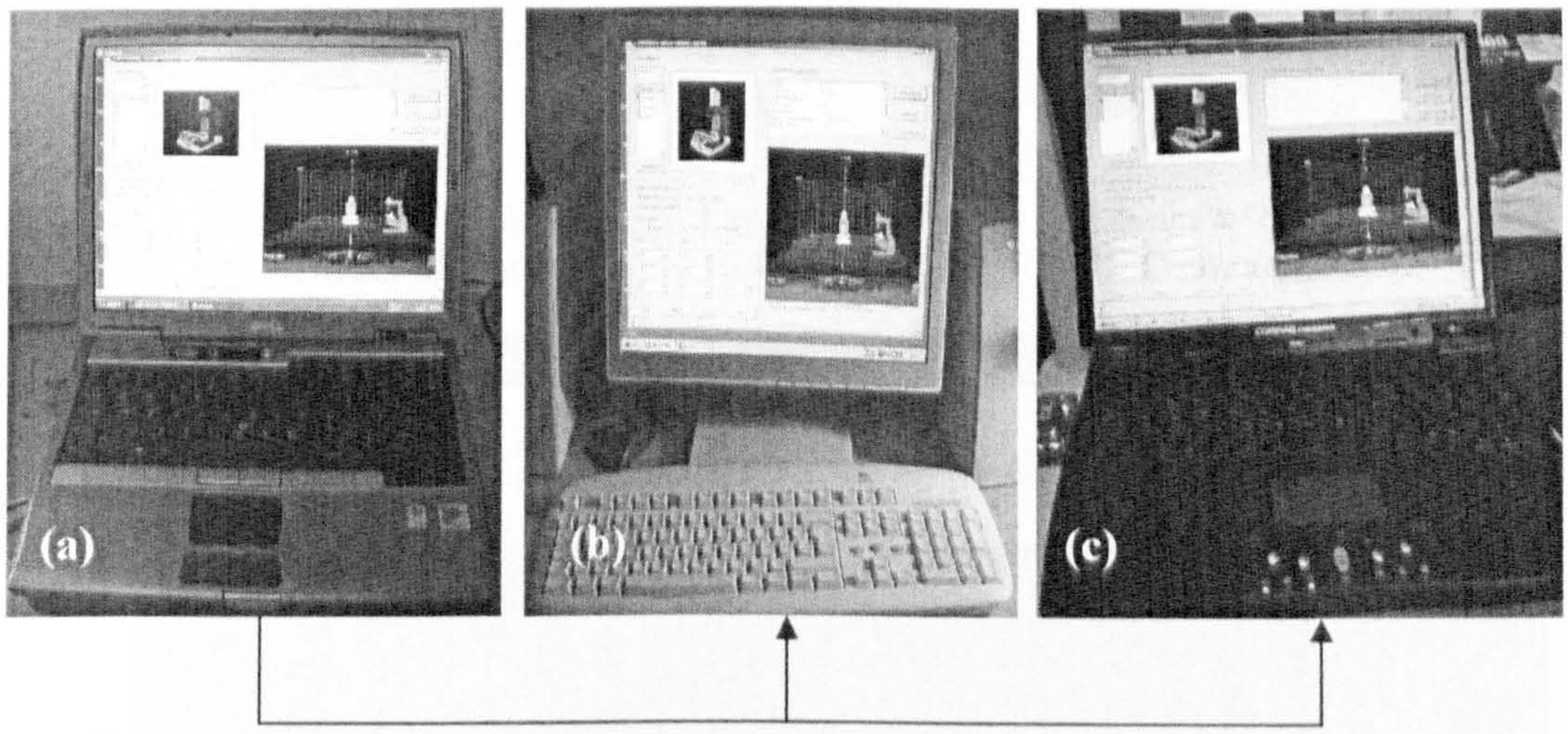


Figure 6-17 The collaboration working experimental example 2– *User2* controlled the VFLS, *User1 and User3* can see the change of virtual scenes on their own machines

Similarly, *User3* controlled the VFLS to add Model3-1 from VMD into the 3D virtual world on **Computer-C** as shown in Figure 6-18 (a). And *User1* and *User2* then saw the change of virtual scenes on their own machines (**Computer-A** and **Computer-C**) as shown in Figure 6-18 (b) and (c).

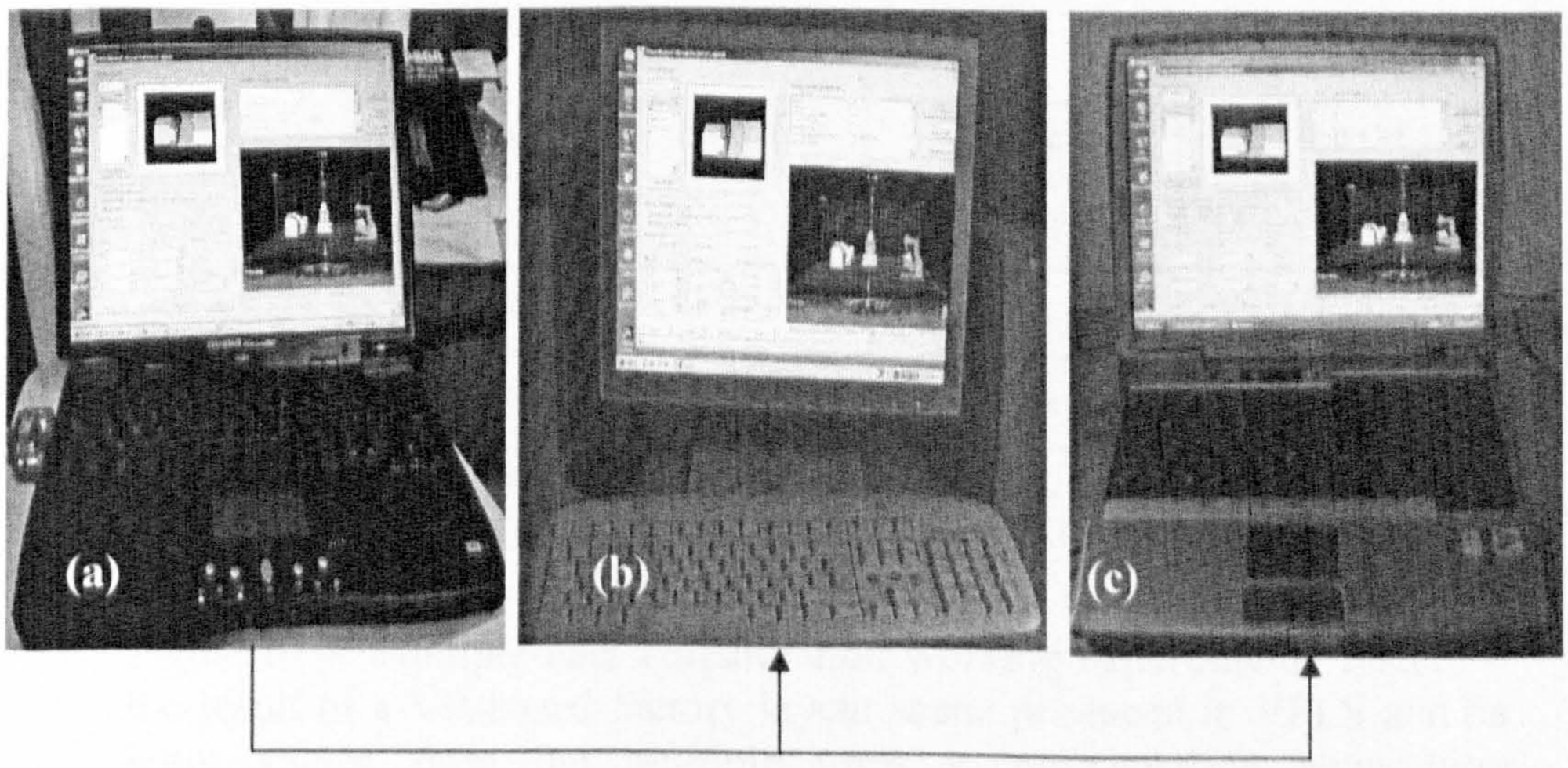
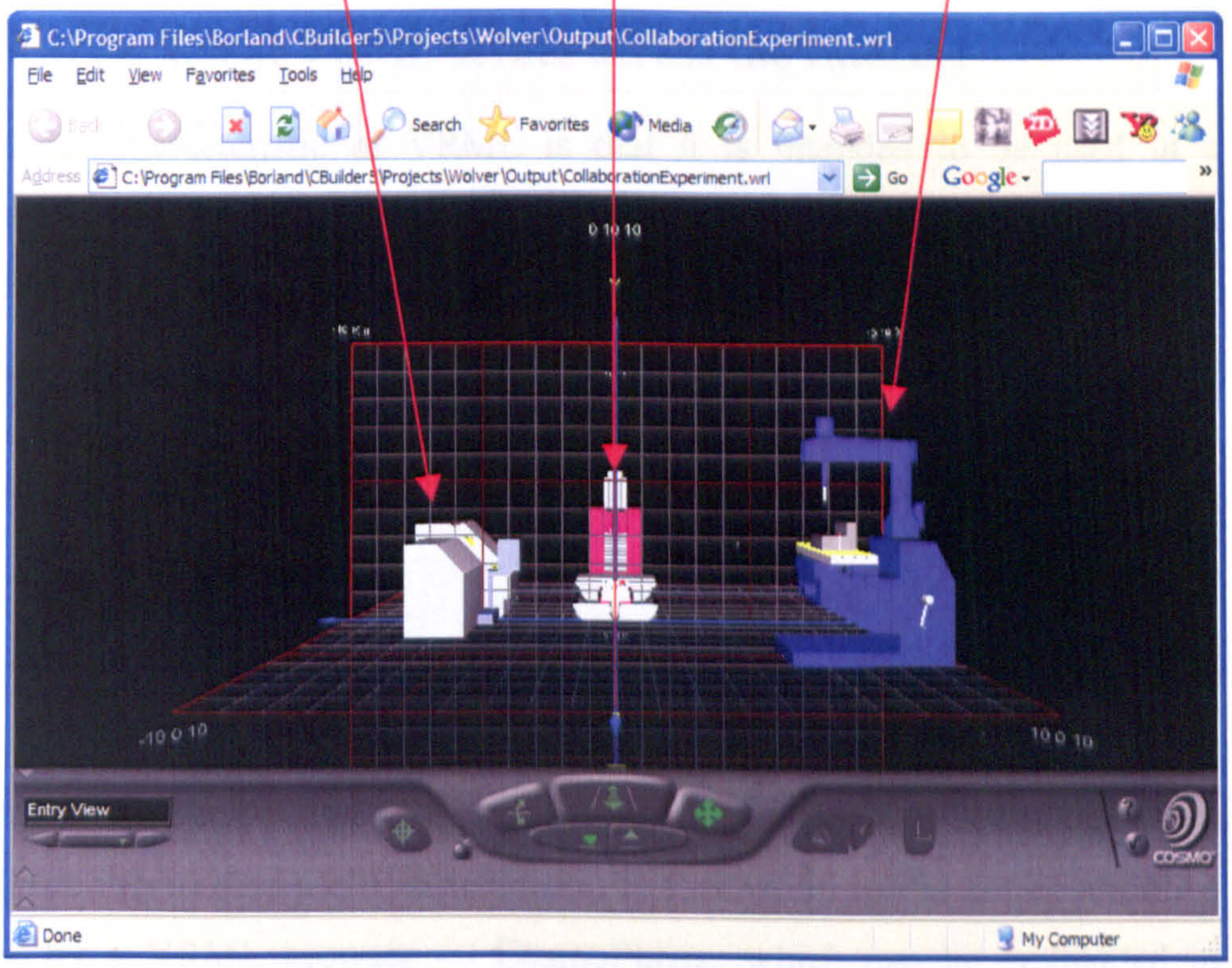


Figure 6-18 The collaboration working experimental example 3 – *User3* controlled the VFLS, *User1 and User1* can see the change of virtual scenes on their own machines

As a result, a VR-based factory layout scene was successfully produced in the VFLS by the multiple users via collaboration working in the networking environment. Figure 6-19 shows the result of collaboration working experiment implemented by multiple users.



Controller	Model	Position (x, y, z)	Size (x, y, z)	Orientation (Angle & axis)
User1	Model1-1	(5, 0, 5)	(2, 2, 2)	-90°, Y-axis
User2	Model2-1	(0, 0, 0)	(1.5, 1.5, 1.5)	0°
User3	Model3-1	(-5, 0, 0)	(1.5, 1.5, 1.5)	90°, Y-axis

Figure 6-19 Multiple-user collaboration working experimental result — the result of a VR-based factory layout scene produced in VFLS and its input values from the multiple users in collaboration networking environment.

The collaboration working experiments have been done successfully by the author not only on the local area network but also over the Internet in this research project. The experimental results show that the VFLS developed by the author in this research project can efficiently support multiple users for collaboration working in design and manufacturing over a local or wide area network.

6.2.4 Distribution of VRML scenes across the Internet

An essential characteristic of VRML is that it is intended to be used in a distributed environment such as the World Wide Web. There are various mechanisms built into the language that support multiple distributed files, including: in-lining of other VRML files; hyperlinking to other files; using established Internet and ISO standards for other file formats; defining a compact syntax.

In this project, the combination of these mechanisms in the creation of VRML scenes allows the Web-based users to get benefits from VR technology in a distributed environment. After VRML senses were produced in the VFLS, they have been delivered onto the Web server via FTP program by the author and then displayed by a VRML browser — CosmoPlayer, which can be embedded in normal HTML-based Webpages.

As a result, the user distributed in the different places is able to remotely access, navigate, and interact with the 3D scenes in a virtual environment in real time through the Internet. Figure 6-20 shows the Webpage in which a VRML-based factory layout scene was embedded by the author. It can be accessed at http://vitalstatistix.nicve.salford.ac.uk/ ~lijin/ VFL/layout.htm*.

* The author worked as a Research Fellow in the Centre for Virtual Environments, University of Salford from 2002 to 2003. Thus, the experimental results were uploaded onto the Web server of University of Salford for access and testing.

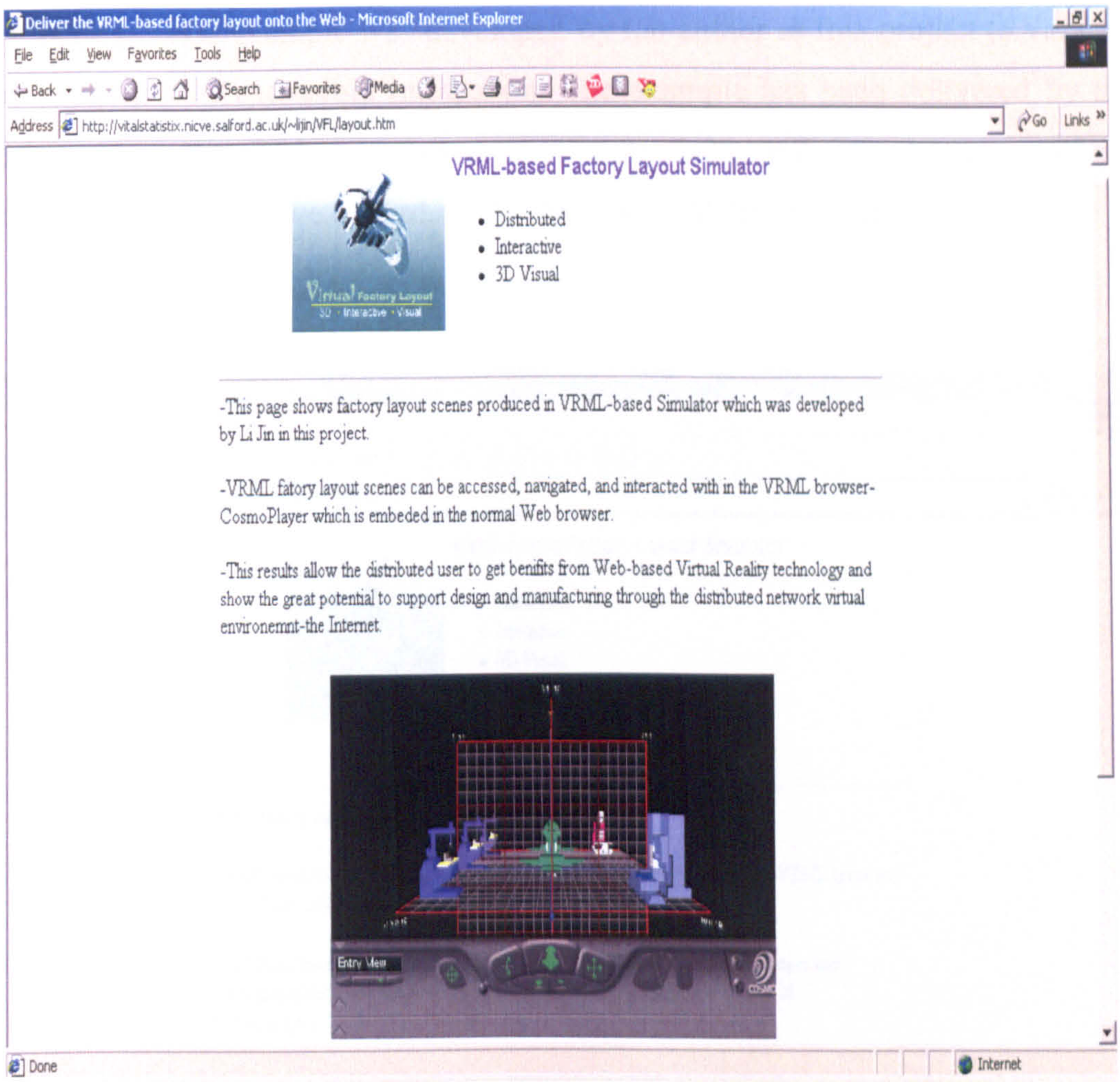


Figure 6-20 A VRML scene produced by the VFLS was embedded into a normal Web browser – Internet Explorer (developed by the author)

6.2.5 Integration of HTML-based information systems

After VRML scenes are distributed across the Internet, there is a requirement that to establish the relationship with other relevant information systems on the Web. Since VRML supports ***hyperlink*** mechanism, VRML scenes are able to hyperlink to other HTML-based files and even information systems through the Web. The section describes how a VRML-based scene can become a 3D interactive interface of an extensible synthetic information system by taking advantage of this mechanism.

An experimental example was developed by the author in this project to validate the capability of extensible integration. The example has been delivered by the author onto the Web as shown in Figure 6-21 and can be accessed at http://vitalstatistix.nicve.salford.ac.uk/~lijin/ VFL/VRfactory.htm.

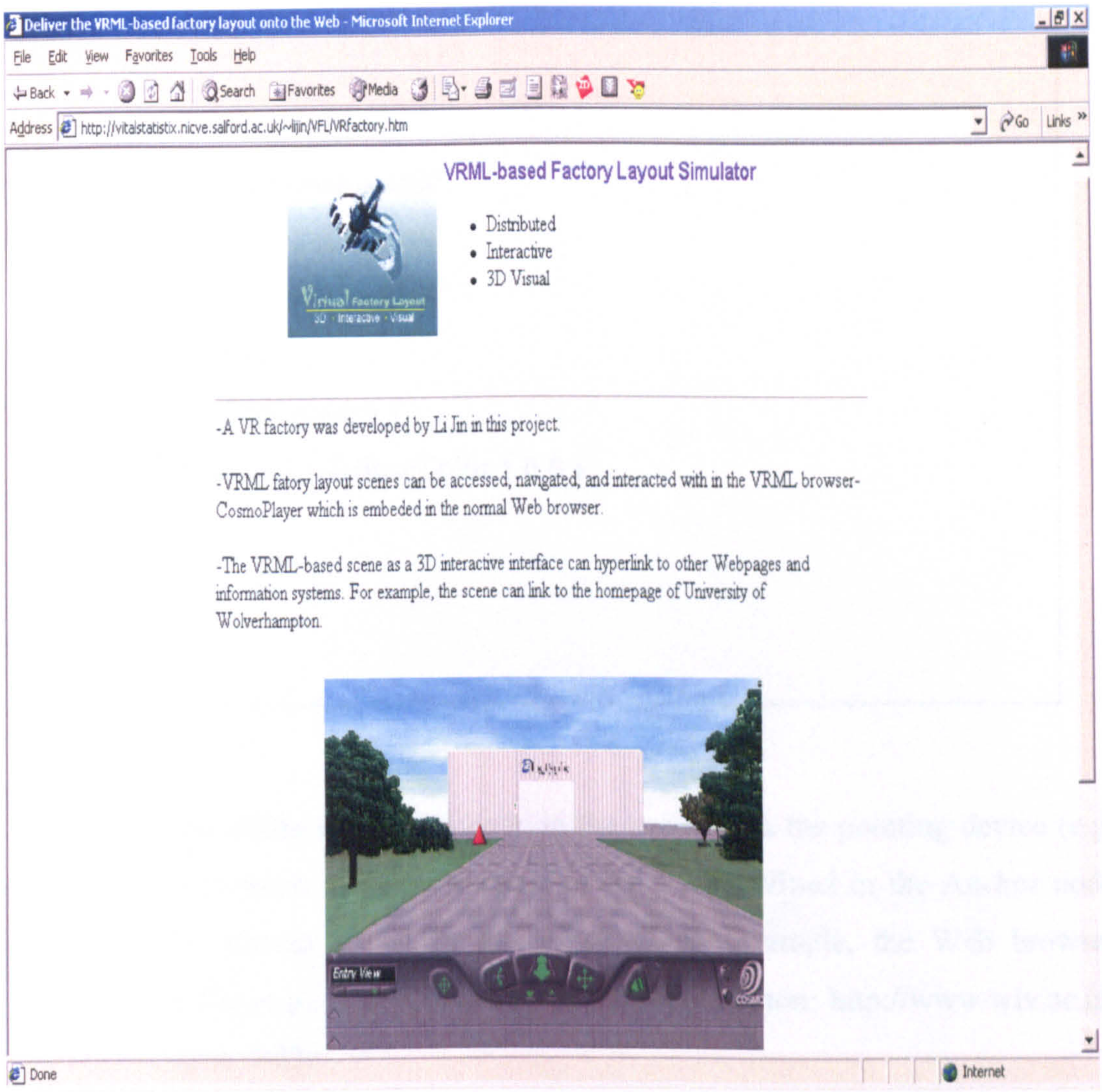


Figure 6-21 An experimental example - an interactive virtual factory embedded into a normal Web-browser (developed by the author)

In VRML2.0, the ***Anchor*** node retrieves the content of a URL when the user activates (e.g., clicks) some geometry contained within the Anchor node's children.

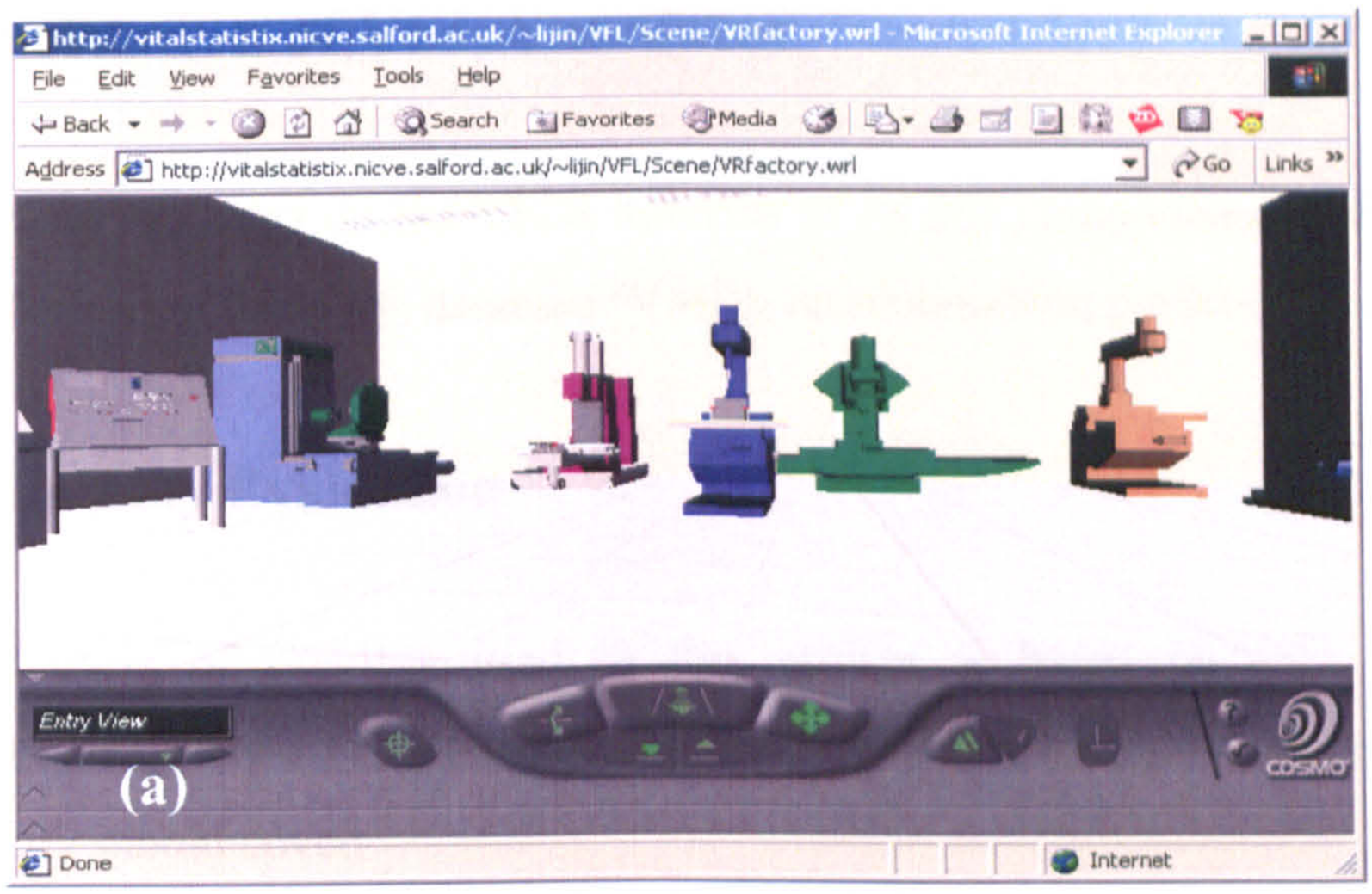
If the URL points to a valid HTML file, the Web browser displays the Web page. As shown in Figure 6-21, there is red cone geometry in a virtual factory scene. An ***Anchor*** node was bound on the red cone geometry by the Author. Its VRML description is as follow:

```
...

Anchor {
url "http://www.wlv.ac.uk"
children
Shape {
geometry Cone { }
appearance Appearance {
material Material { diffuseColor 1 0 0 }
}
}
...
```

When the user clicks on the red cone in the scene with the pointing device (e.g. mouse), it will result in the new scene of the URL defined in the Anchor node replacing the current scene. In this experimental example, the Web browser displays the Home page of University of Wolverhampton: http://www.wlv.ac.uk shown in Figure 6-22c.

Therefore, it is a cost-effective solution to integrate VRML scenes produced in VRML-based Factory Layout Simulator (VFLS) with other HTML-based files (based on text, image, and video files) and even database and information systems through the Web.



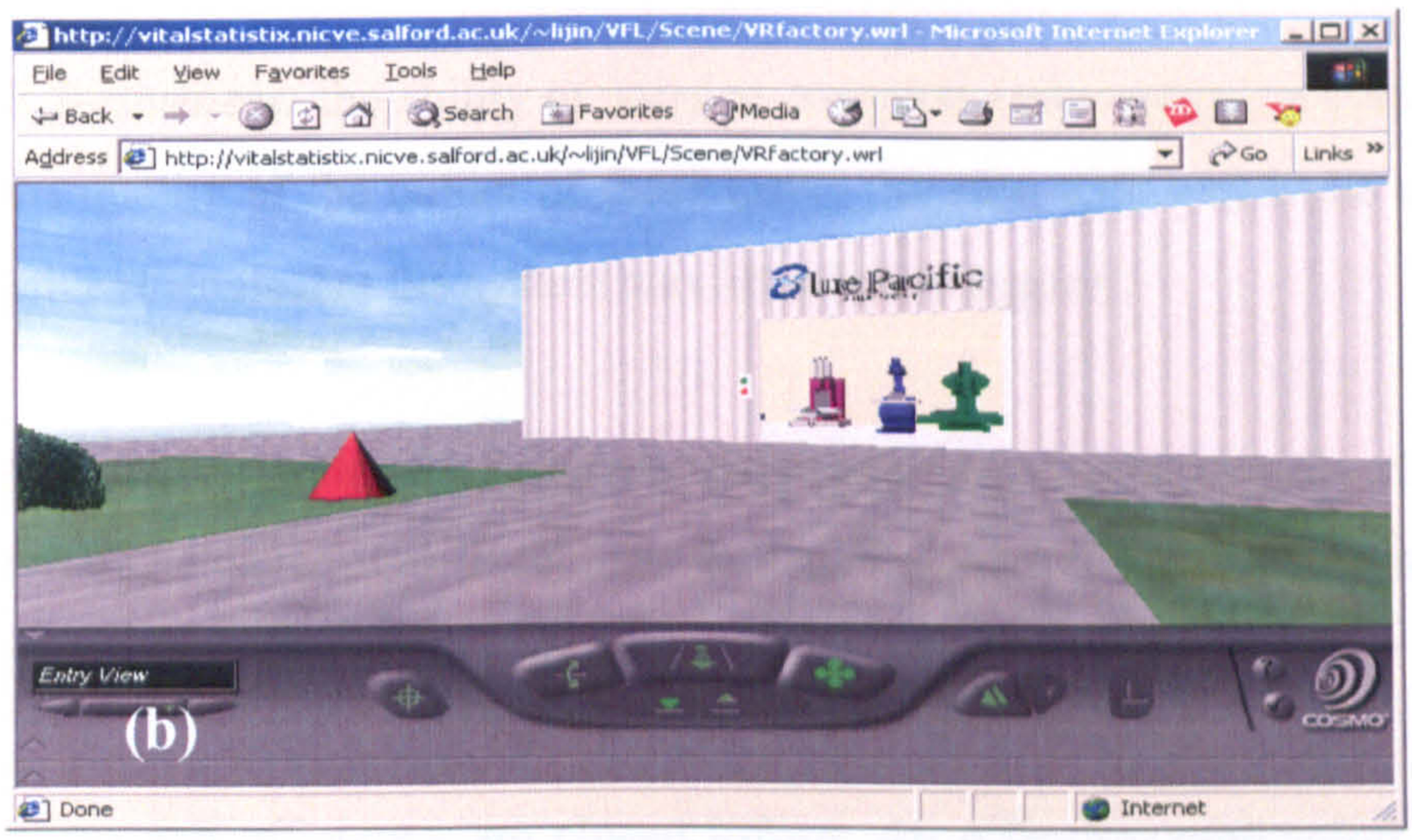


Figure6-22 An experimental example - Integration of HTML information system via the Web. (a)The inside view of virtual factory. (b) The outside view of virtual factory. (c)Hyperlink to University's Homepage (developed by the author)

6.3 Evaluation of VFL Simulator

This section evaluates on the VFLS in terms of its key components. It discusses the evaluations of usability, database (VMD), and interaction performance.

6.3.1 Usability Evaluation

The definition of usability used in this project is based on ISO 9241-11 (International Standards Organisation, 1998). ISO 9241-11 suggests that measures of usability should cover:

- ***Effectiveness:*** the ability of users to complete tasks using the system, and the quality of the output of those tasks, the ease with which the user can achieve goals that the system was intended to support.

- ***Efficiency:*** the level of resource consumed in performing tasks, the goals can be achieved with acceptable levels of resource, whether this is mental energy, physical effort or time.

Effectiveness and efficiency are closed linked. An effective application enables the user to perform their activities in an efficient manner. According to the results shown in Section 6.2.2, the VRML-based Factory Layout Simulator enables the user to produce various factory layout scenes in a cost-effective way. It has integrated ideas and techniques from various disciplines as follow:

- ***Virtual Reality technology:*** offers a virtual simulation environment with the ability of 3D interaction in real time.
- ***Internet technology:*** provides a means of communication and distribution for information sharing and collaboration working through the Web.
- ***CAD technology:*** combines the professional capability of modelling to aid the development of applications in design and manufacturing.

- ***Database technology:*** integrates the ability of database and information management.

Thus, as a demonstrator, the VFLS has successfully validated the proposed Web-based VR approach to support design and manufacturing in this research project.

6.3.2 Virtual Machine Database evaluation

The Virtual Machine Database (VMD) as VR database in the VRML-based simulator has been constructed and described in Section 5.2.2, Chapter 5 – ***Implementation***. The performances of VMD are evaluated as follow:

- **Stability:**
  VMD was constructed as a VR database that integrated geometric data of 3D object and relevant information on design and manufacturing. It has been accessed and used successfully by the VFLS to produce the various layout plans.

- **Maintainability:**
  The structure of VMD is established in ***Database Table*** style. Such table-style database can be maintained easily by directly accessing data and information stored in every record and field.

- **Extensibility:**
  The geometric data of 3D object along with its relevant information as a *record* is stored in rows and columns of the *Database Table* —VM.DB. Thus, it is easy to expend the content of VMD by adding more records into *Database Table*. These new records will contain new 3D objects' data and information. In addition, the VMD can be evolved into customised VR database according to the user's requirements for specific applications.

6.3.3 Interaction performance experimental evaluation

The interaction performance is directly affected by the complexity of a virtual scene. This section evaluates the interaction performance of VRML-based factory layout scenes with different level of complexity. The complexity of a virtual scene depends on the number of 3D objects or the overall polygonal counts involved in the virtual environment. For the sake of simplification, the project uses the same object (Model1-1) in virtual scenes to evaluate the interaction performance in experiments.

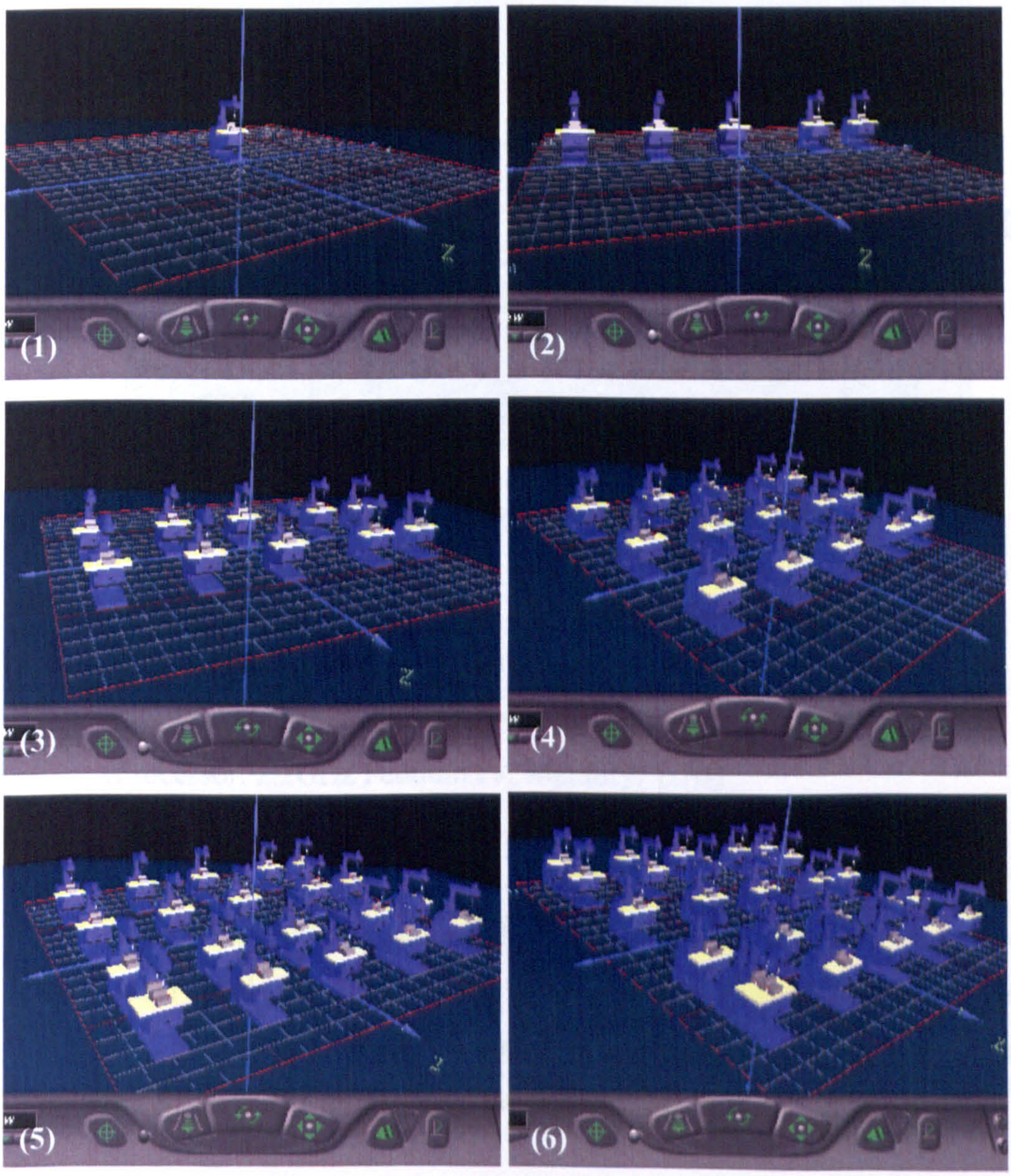


Figure 6-23 The VRML-based factory layout scenes with different level of complexity. (1) 1 object, (2) 5 objects, (3) 10 objects, (4) 15 objects, (5) 20 objects, (6) 25 objects. (developed by the author)

Figure 6-23 (1)-(6) respectively contains 1, 5, 10, 15, 20, and 25 models of the milling machine. Obviously, the level of complexity of VRML scenes is increased by the number of 3D objects in the scenes. The interaction performances of these VRML scenes have been compared by testing their rendering time in experiments and summarised in Figure 6-24.

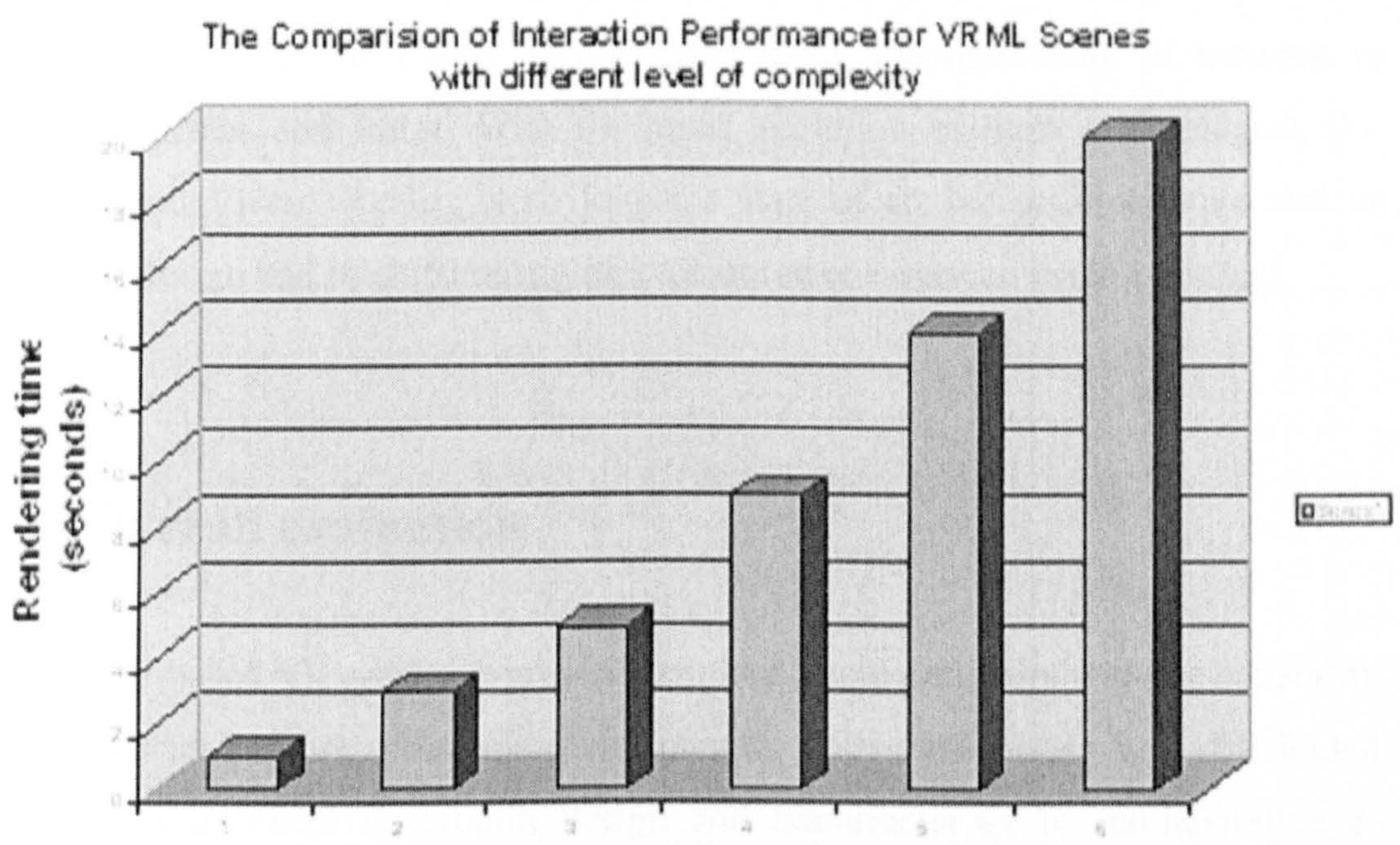


Figure 6-24 The comparison of interaction performance for the VRML-based factory layout scenes with different level of complexity. The experiment was done by the author on the PC (Processor: 2.2GHz Pentium IV; Memory: 1GB)

According to the testing results as shown in the above chart, a complex scene takes longer rendering time than a simpler scene does. With the increasing rendering time, the interaction performance of a virtual environment decreases due to the delay of real-time display latency.

In addition, since the VRML scenes are intended to be delivered across the Internet, the more complex a VRML scene is, the longer its download time is.

Meanwhile, the interaction between the Web-based user and the virtual environment becomes slow due to the limitation of network bandwidth. In order to address this problem, it is necessary to compromise the complexity and interaction performance of VRML scenes in order to obtain an acceptable navigation speed and interaction performance. Many research works on polygon reduction and optimization have been done to decrease the complexity of 3D objects (Lubke et. al. 2002), on the other hand, the bandwidth of network is growing faster and faster. With the rapid evolution of these technologies, the Web-based Virtual Reality is no longer a state of art but an innovative tool to support design and manufacturing for collaborative e-service via the Internet.

6.4 Overall evaluation

The Web-based VR approach provides engineers and designers with the capability to visualise, explore, manipulate and interact with applications in a distributed virtual environment to support design and manufacturing for collaborative e-service via the Internet. The section discusses the overall evaluation of approach proposed by the author in this research project. Compared to the conventional CAD/CAM approaches to design and manufacturing (e.g. Deneb's products – Virtual NC and QUEST), the proposed Web-based VR approach and the generic framework of prototyped VRML-based simulator in this research project have the following advantages:

- ***3D visual:*** Quickly produce various VRML scenes in visual 3D way by manipulating 3D objects.

- ***Real-time interaction:*** Allows distributed Web-based users to interact with the virtual environments in real-time.

- ***Distributed:*** Deliver 3D visual virtual environments across the Internet.

- ***Extensible database:*** Permits sharing a large database on a server among distributed users through the WWW.

- ***Reuse resource:*** Be able to reuse the available software sources such as CAD packages.

- ***Standardization:*** Be compatible with Web-based VR international standard (ISO/IEC 14772-1:1997) —VRML.

- ***Platform-independence:*** Performs on a wide variety of computing platforms. Therefore, distributed users (e.g SMEs) can take advantage of their already existing computer facilities.

- ***Low-cost:*** No licensed software and additional system maintenance are required for distributed users.

- ***Scalability:*** Handles large numbers of distributed users in a large-scale virtual environment.

- ***Extensibility:*** Enables arbitrarily large dynamic virtual environments and hyperlinks to other engineering information systems according to the user requirements, especially for SMEs.

- ***Collaboration:*** Supports multiple-user access. For example, a cluster of co-operated SMEs could communicate with each other for collaboration working.

In summary, the generic VRML-based simulation system based on this approach will accelerate the major activities of engineering, including: manufacturing process concept development and simulation, optimising assembly lines and workshops design, integrating labour and equipment, etc.

6.5 Conclusions

This chapter presented the software toolkit — VRML-based Factory Layout Simulator as a demonstrator and described results in terms of its functionality and performance. The content of VR Database — Virtual Machine Database (VMD) has been examined, and then various factory layout scenes have been produced by this interactive simulator as examples to validate the usability of the software toolkit. Furthermore, this chapter described the result of collaboration working experiment among multiple users. The collaboration working experiment has successfully validated that the Web-based VR software system developed by the author can efficiently support multiple users for collaboration working in design and manufacturing over a local or wide area network.

In addition, the chapter described the final distribution of VRML scenes across the Internet and the integration of HTML-based information systems through the Web. Finally, the chapter carried out the evaluations on the VFLS in terms of its database and interaction performance and the overall evaluation of the Web-based VR system proposed in this project.

Chapter 7

Conclusions and Future Work*

7.1 Summary and conclusions

With the rapid growth of network technology and the development of Web-based VR techniques, there is a potential opportunity to extend low-cost VR applications and distributed systems via the Internet for collaborative e-service. However, according to a literature review of previous research work on VR, it has been shown that little research work had been done in applying the Web-based VR technologies to support design and manufacturing. The aim of this research project was to investigate Networked Virtual Environments (Net-VEs) and propose an approach to apply Web-based VR technologies to support interactive applications in design and manufacturing for collaborative e-service.

This thesis has proposed a cost-effective approach along with the detailed architecture to apply Web-based VR to support applications in design and manufacturing. Based on the proposed approach, a distributed VR system has been designed by the author using VRML. Furthermore, the thesis also provided the solutions for key components involved in the Web-based VR system including geometric modelling for creating VR database, dynamic simulations in VR

* Part of this chapter has been published in *Web 3D*, edited by S. Dredge, pp. 70-75. ISBN 185669-283-3, Lawrence King Publishing Ltd, London, 2002.

engine, and network communication. A prototype software system — the distributed VRML-based Factory Layout Simulator (VFLS) for factory layout applications has been successfully developed as a demonstrator to validate the viability of the proposed approach and the usability of the distributed VR system. It is capable of aiding multiple users, especially SMEs to carry out engineering activities in a networked virtual environment at lower cost and in less time and sharing design and manufacturing resources (e.g. database and information systems) through the World Wide Web (WWW) without any extra economic burdens.

The following items summaries the research work that has been carried out by the author.

1. Literature review on VR technology, Net-VEs and previous research and applications have been carried out.

2. A cost-effective approach using Web-based VR technologies to support interactive applications in design and manufacturing has been proposed for collaborative e-service via the Internet. The detailed architecture of the distributed VR system by using VRML has been designed based on the proposed approach.

3. The key components involved in the distributed Web-based VR system — VR database, VR engine, and network communication have been explored and developed. Critical integration issues concerning 3D modelling for creating the VR database, dynamic simulations in the VR engine, and network communication have been discussed and developed.

4. A prototype software system — the VRML-based Factory Layout Simulator (VFLS) for factory layout applications has been developed by the author using UML in order to demonstrate the proposed approach and framework.

5. The major implementation components of the prototype system include: the construction of VMD; the creation of grid-enhanced 3D coordinate system; the implementation of "*Inline*" mechanism for the creation of complex VRML scenes; the composition of 3D transformation for layout operations; and the introduction of '*Virtual Sensors*' for real-time interactive manipulation. The final distribution of VRML-based virtual scenes across the Internet and the integration of HTML-based information systems have been carried out through the WWW.

6. As a demonstrator, the implementation of VFLS has successfully validated the viability of the proposed approach and the usability of the Web-based VR system.

7. The results and evaluations of the VFLS in terms of its database — VMD and interaction performance and the overall evaluation of a proposed Web-based VR system have been carried out.

8. The multiple-user collaboration working experiments have been done successfully by the author. The experimental results show that the VFLS developed by the author can efficiently support multiple users for collaboration working in design and manufacturing over a local or wide area network.

In summary, this research work has contributed the research effort in using Web-based VR to support interactive applications in design and manufacturing, to bridge the gap between VR and distributed applications through the Internet. This research work proves that a distributed virtual sharing environment based upon the WWW could allow industries to utilise VR sufficiently and cost-effectively. The proposed approach and system will enhance and accelerate the major activities of engineering, including: product concept development and simulation, manufacturing process, optimising assembly lines and workshops design, integrating labour and equipment, etc.

7.2 Limitations and enhancement

During the period of the project development, Virtual Reality Modelling Language (VRML) as a key technology has become an International standard for describing interactive 3D scenes delivered across the Internet/Intranet. It adds the 3D interaction and immersion to the online experience. However, since VRML does not define a visual Application Programming Interface (API), this shortcoming not only limits the VRML application developer in creating a larger and more complex VR applications efficiently but also especially restricts less programming technical users in creating a VRML world. Recently, more Internetworked 3D graphics techniques (Rhyne et. al. 1999) such as MPEG-4 (Koenen 1999) and the Extensible 3D (X3D — next generation of VRML) standard began to emerge. At the final stage, this research project has attempted to integrate with other Web-based VR technology and new platforms such as Adobe Atmosphere. This section explores a new Web-based VR tool — Adobe Atmosphere as enhancements in this research project.

Increasingly, researchers and companies have seen the great potential of Web-based VR for distributed interactive applications. The Adobe Company has begun to develop a new product called "Atmosphere". It aims to provide a high-level, interface user friendly tool for common users (e.g. artists, architects) to build and explore interactive 3D contents on the Web. The new released Adobe Atmosphere 1.0 (Spring 2002) was designed for Web-based 3D interactive applications (Adobe Atmosphere 2003).

In interests of the high-level and easy-use of Adobe Atmosphere, the project has explored the software package as an additional solution for Web-based interactive applications for design broadcasting such as product design and architecture reviewing, online education and so on.

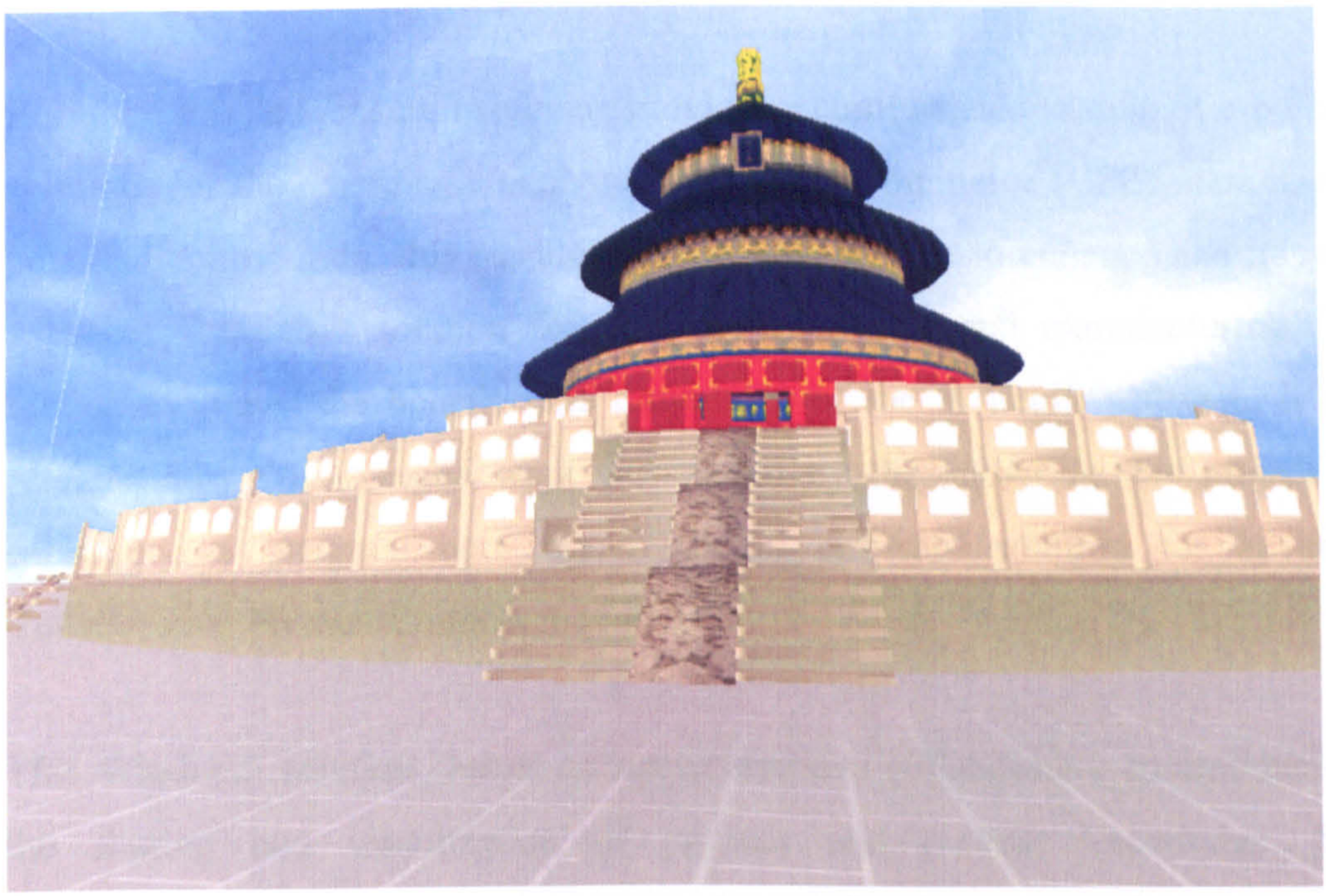


Figure 7-1 A Web-based virtual world - the Temple of Heaven, Beijing, China for architecture reviewing developed by the author using Adobe Atmosphere.

As a trial, this project has developed a Web-based virtual world in Atmosphere to visualise a famous Chinese royal architecture — the Temple of Heaven*, which is located in Beijing, China. It was used to worship heaven by emperors for prosperity and peace 580 years ago. The Atmosphere virtual world shown in Figure 7-1 successfully represents the best-known and most splendid architecture — Qi Nian Dian (Hall of Prayers for Good Harvests) on the Web in 3D visual and interactive way. It allows exploring the architecture with Chinese classical music with a sense of presence in Adobe Atmosphere Browser. It can be accessed by multiple users distributed in different locations. The website is at http://www.adobe.com/products/atmosphere/contest.html.

* The author was awarded the **First Prize** — ***the Most Creative and Immersive Web-based Virtual World*** by Adobe Company in 2002.

7.3 Enhancing and integrating with Windchill® solution for advanced Collaborative Product Commerce

Recently, PTC's Windchill software provides a comprehensive suite of e-business solutions for the emerging Collaborative Product Commerce (CPC) market. As a potential contribution, this research project will be able to enhance and integrate with the **Windchill** solution for e-service in design and manufacturing. This section evaluates the benefits and limitations of the Willchill software. It then presents how this research work carried out by the author in this project can enhance and integrate with the Windchill software solution for advanced Collaborative Product Commerce (CPC).

This Windchill solution creates an inter-enterprise collaborative environment for the sharing and visualization of product and process knowledge. Such collaboration system transforms product knowledge into a significant enterprise business asset by making it available to the extended enterprise. The Windchill's integrated suite of application modules that address the requirements for product and process lifecycle management on a federated architecture is shown in Figure 7-2.

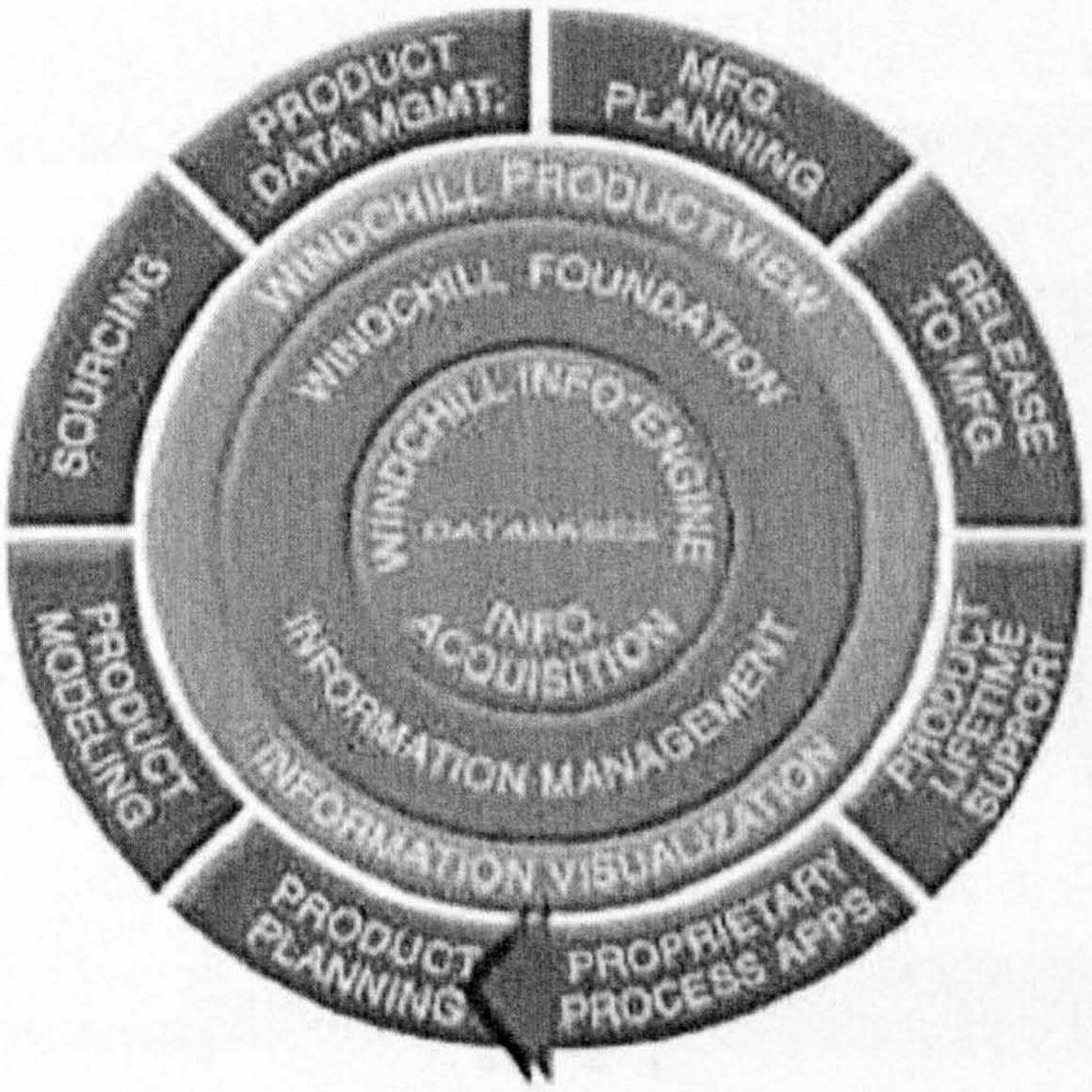


Figure 7-2 The Windchill's integrated suite of application modules (PTC 2003)

The core of **Windchill®** collaboration system consists of **Windchill Foundation**, **Windchill Info*Engine**, and **Windchill ProductView**.

- ***Windchill Foundation*** supports the entire Windchill solution. It allows access to the most current information, and security systems, which manage access to product information based on the product's lifecycle stage.

- ***Windchill Info*Engine*** is an enterprise application integration framework that allows companies to create business applications comprised of information extracted from multiple data sources through a Web-based composite application.

- ***Windchill ProductView*** provides CAD-independent viewing capabilities. It allows users to access nearly any type of product-related information, ranging from online forms and applications to documents, drawings, and 2D and 3D solid models. Users can view the various types of 2D and 3D standard format to hold online conceptual reviews of new product design as shown in Figure 7-3.

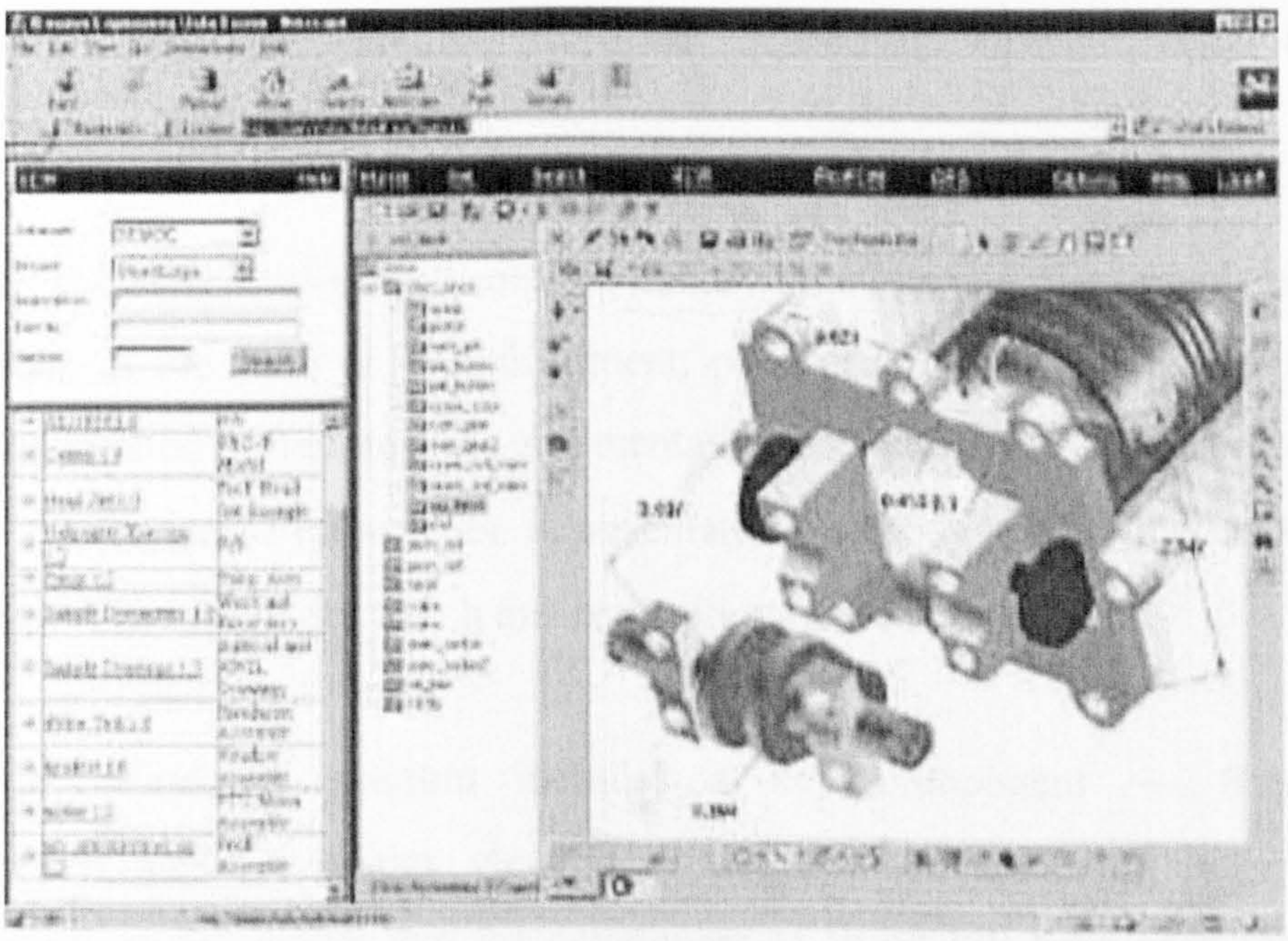


Figure 7-3 The interface of Windchill's ProductView

The benefits of the Windchill collaboration system include:

- It creates a strategic source of product and process information and presents it in a collaborative environment where all members of the value chain can communicate. Just like the Web, this is done independently of where the information is located or what application it was created on.

- It enhances the value of existing IT investments by connecting and leveraging information among business systems.

- It facilitates inter-enterprise collaboration which allows customer, supplier, and partner collaboration to deliver innovative products through the Internet.

- It makes manufacturing companies be able to respond to customer demands for products that address particular requirements. Suppliers and partners can provide domain expertise not available within the company.

At present, many well-known companies and organisations such as NASA, BMW and Volvo have got benefits from the Windchill software solution for collaboration e-business.

However, the current Windchill solution mainly transfers and distributes data and information in the form of text, document, product structure and images. Such traditional methods of information representation present severe limitations due to the lack of innovative information representation methods on the Internet — the Web-based Virtual Reality which this research project is targeting.

Since the Windchill solution includes a key component — ***Windchill ProductView*** which provides viewing capabilities for achieving information visualisation, it has shown that the developers of Windchill have seen the great potential of Web-based 3D technology for the information visualisation on the

Internet. However, the current functionality of ***Windchill ProductView*** just offers the user to simply view the 3D model of a product and still lacks real-time interaction between the user and the 3D models on the Web. Such limitations result in low efficiency and performance of the information representation and learning process. This evidence has shown that the Windchill software solution has not adopted and sufficiently incorporated the Web-base VR for advanced interactive information visualization so far although the developers of Windchill have begun to make effort in this direction.

The evolution of Collaborative Product Commerce (CPC) has aroused a demand for the development of more effective information representation tools and methods to overcome such limitations. Virtual Reality, as a powerful tool of information visualization, provides a better understanding of information and contents by the direct interactive manipulation of data and information and through the creation of safe simulation platform and environments for training, learning, and maintenance. It will enhance e-service and CPC by providing advanced interactive solutions rather than simply transferring text and document information through the Web.

Thus, the Web-based VR approach proposed and discussed by the author in this thesis as an innovative information representation method will be able to contribute to enhance the functionality of ***ProductView*** in the Windchill software system for advanced Collaborative Product Commerce (CPC). By taking advantage of real-time interactive manipulation feature of Web-based VR in product visualization and design, users will increase productivity, speed decision making, reduce costs, and improve time to market by eliminating costly and timely travel aspects of the review processes. It further improves communications between engineers, clients, suppliers, and product teams prior to product launch, and reduces the need for multiple revisions and prototypes.

Furthermore, since the Windchill software is based on a Web-based infrastructure with a federated architecture, it is compatible with other Web-based systems and

applications. Theoretically speaking, the Web-based VR system developed in this research project can seamlessly integrate with the Windchill software to support collaboration e-service in design and manufacturing. Further research needs to be carried out to investigate and implement such integration in the future work.

As companies must be able to respond quickly to their dynamic business environment, such enhanced integration system can support a collaboration environment by providing companies with access to the product and process information they need to make the best business decisions, and this allows them to identify new market opportunities and drive new levels of competitiveness.

7.4 Future work

In summary, the development of the next generation of Web-based interactive VR applications requires effective integration of various Internetworked 3D graphics techniques (Rhyne et. al. 1999), the emerging new tools as Adobe Atmosphere, and the Collaborative Product Commerce (CPC) solution such as PTC's Willchill. Using standard Web browsers as an execution engine for interactive VR applications, it enables common users to remotely access, share, and interact with data and information represented in 3D computer-generated virtual environments at different global sites via the Internet.

The School of Engineering and the Built Environment at the University of Wolverhampton established the Innovative Product Development Centre (IPDC) at Telford, UK. The centre aims to assist SMEs to become more competitive by sharing design and manufacturing resources through the WWW. The centre includes a digital design and manufacturing lab and a manufacturing equipment workshop. Figure 7-4 shows the overview of the IPDC.

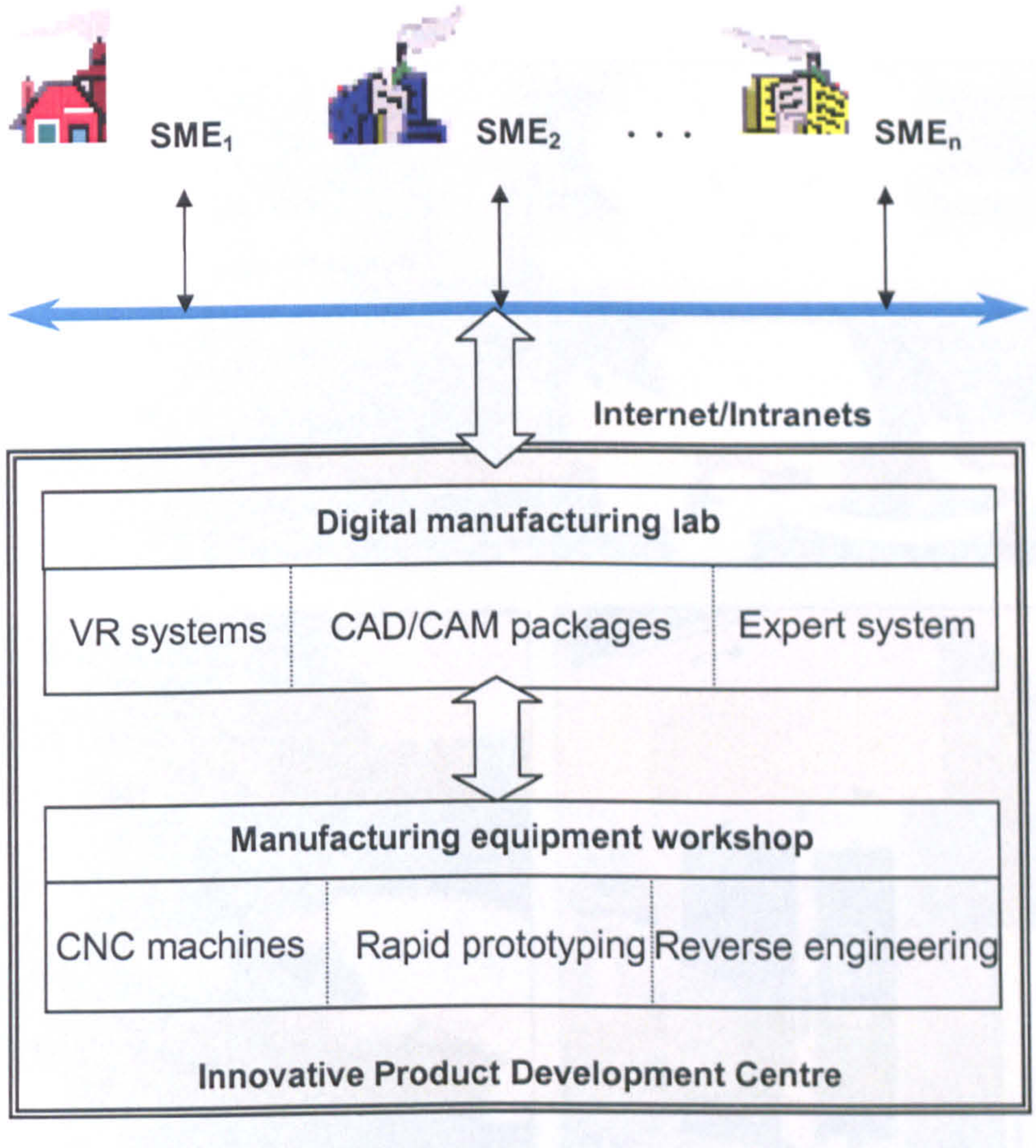


Figure 7-4 The overview of the IPDC, Telford, UK.

The digital design and manufacturing lab consists of VR systems, several common commercial CAD/CAM packages such as PTC's Pro/E, Autodesk's AutoCAD, Deneb CAD/CAM system and so on, and a knowledge-based expert system for product assessments. The manufacturing equipment workshop contains latest equipment shown in Figure 7-5 such as a rapid prototyping system, reverse engineering instruments, high-speed Computer-Numerical-Control (CNC) machines and so on.

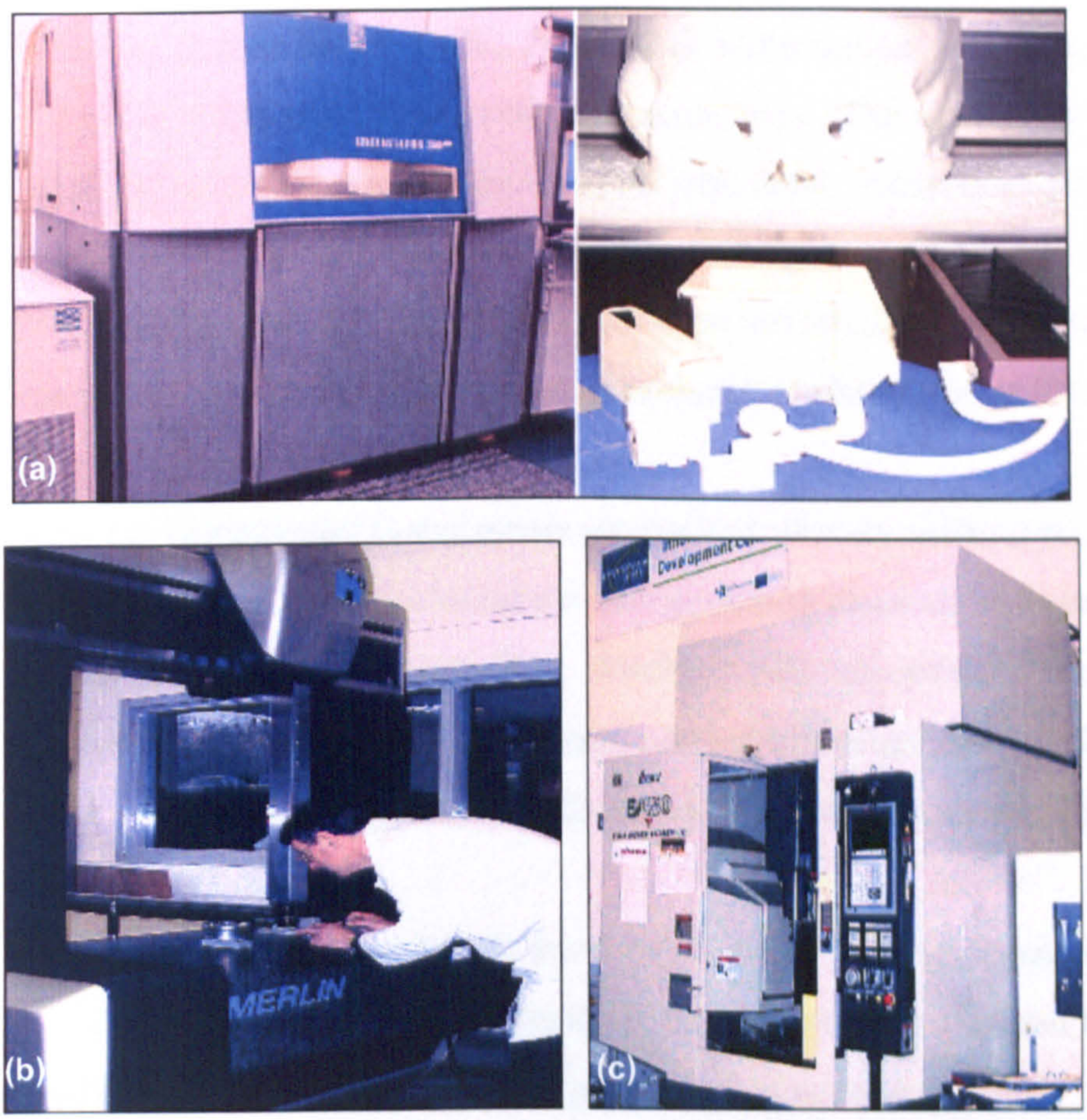


Figure 7-5 The views of latest manufacturing equipment in the IPDC.
(a) The rapid prototyping system—DTM SLS Sinterstation and some product prototypes produced for SMEs.
(b) The reverse engineering instrument—IMS Merlin CNC co-ordination measuring machine.
(c) The multi-axis CNC machine.

- **Rapid Prototyping system** produces accurate physical models quickly, safely and automatically. It gives CAD designers the freedom to quickly create and hold a physical 3D model to handle. It brings a new product from a concept design in virtual reality to an actual object in physical

reality. It enables the production of small to medium-sized prototypes; precision builds of extremely detailed small and fine parts; and rapid creation of complex geometry and feature-dense parts etc. The laser sintering station in the CMC, DTM SLS Sinterstation 2500plus, uses a plastic or nylon powder to create prototype parts. The powder is melted, layer by layer, by a computer-directed heat laser. Additional powder is deposited on top of each solidified layer and again sintered. The process provides the most functional rapid prototype available. Figure 7-3a shows the station along with some product prototypes produced for SMEs.

- **Reverse engineering instruments** interpret an already existing product by analysing the design considerations that governed its creation. Figure 7-3b shows the most advanced IMS Merlin CNC co-ordinate measuring machine in the CMC. With a 3D non-contact scanning head, it allows all surface features to be quickly digitised and transported to a CAD file.

- **High-speed CNC machines** have high-speed cutting capability and spindle speeds; efficient swarf removal; fast in-cycle load/unload and part positioning; super-fast tool changing and positioning times. Client's CAD/CAM programs can be imported directly into CNC machines and then run at high speeds. The center houses a variety of four and five axis CNC machines. Figure 7-3c shows a multi-axis CNC machine.

Generally, it is impractical and unnecessary for every SME to purchase all of the above expensive manufacturing resources. Thus, we intend to allow registered SMEs to remotely access the IPDC via Internet/Intranet for sharing resources. A client could send a product design along with its specification to the IPDC through the Web. After the design has been modified and optimised by experts in this centre, the actual prototype can be produced for testing and evaluation. Then the final accepted version is converted into the CNC code for performing the manufacturing process or is sent back to the client. During the procedure, designers can discuss their ideas and requirements with experts by network

communication software toolkits (e.g. Microsoft Messenger, email, FTP or networking conference) for collaboration working. At present, the IPDC has begun to serve SMEs in West Midlands—the famous Black Country, the original place of the industrial revolution in history for aiding their product development. Our ultimate goal is to benefit more SMEs on the global marketplace by sharing resources and collaboration working in such a distributed virtual design and manufacturing environment.

In future work, the research work presented in this thesis and the emerging CPC solution such as PTC Windchill software will need to be further incorporated into the IPDC environment. It will become a main component as a part of VR systems for the Internet access. Since most SMEs do not need the full bandwidth of immersion, the proposed Web-based VR system as a cost-effective VR approach for SMEs will be able to aid collaborative product development via the Internet.

In addition, in reality, people always play an essential role in every practical design and manufacturing activity. Therefore, people such as workers beside an assembly line need to be represented in a virtual environment in some applications. In particular, with the introduction of people models to a workshop layout application, a layout design can be enhanced according to ergonomics rules. Such enhancement will aid the improvement of work efficiency and the achievement of the full performance of equipment (Jung and Milde 1999). However, the integration of 3D people may result in overloading polygon rendering on the limited network bandwidth because of its large data. Since the proposed VR-based simulation system is intended to be distributed across the Internet, the navigation speed must be considered to keep the sense of presence. Thus, more effective method needs to be explored to optimise the labour integration in the Web-based VR system in future work. Furthermore, multimedia issues such as video and network meeting modules could be incorporated to improve real-time discussion for collaboration working in order to enhance the Web-based system on a broad base of manufacturing and industrial applications for e-service via the Internet.

Publications and Awards by the Author

Li Jin, (supervised by Ilias A. Oraifige and Frank R. Hall), "The Interactive Web-based VR - Temple of Heaven," was published in *Web 3D*, edited by S. Dredge, pp. 70-75. ISBN 185669-283-3, Lawrence King Publishing Ltd, London, 2003.

Li Jin, (supervised by Ilias A. Oraifige and Frank R. Hall), "Using Web-based VR for interactive 3D applications — the Temple of Heaven" was awarded the First Prize for ***Most Creative and Immersive Web-based Virtual World*** by Adobe company in Jan. 2002.

Li Jin, Ilias A. Oraifige, Frank R. Hall and Paul M. Lister, "E-manufacturing in Networked Virtual Environments" was published in the proceedings of IEEE SMC 2001, IEEE Systems, Man, and Cybernetics International Conference, Oct. 7-10, 2001 at Arizona, U.S.A. (***This paper was award the Best Student Paper Finalist.***)

Li Jin, Ilias A. Oraifige, Frank R. Hall and Paul M. Lister, "Distributed VR-based Simulation for Manufacturing" was published in the proceedings of 13th European Simulation Symposium: Simulation in Industry, Oct. 18-19, 2001 at Marseille, France.

Li Jin, Ilias A. Oraifige, Frank R. Hall and Paul M. Lister, "Distributed Virtual Manufacturing for SMEs" has been submitted to the journal of IEEE Computer Graphics and Applications. (Under review)

Li Jin, Zhigang Wen, "Adorning VRML Worlds with Environmental Aspects" was published in the Journal of IEEE Computer Graphics and Applications, vol. 21, no. 1, Jan./Feb. 2001.

Li Jin, Q. Peng, Frank R. Hall and Paul M. Lister, "Using Web-based VR Technology to Support Design and Manufacturing" was published in the proceedings of the 6th CACS Conference, Loughborough University, UK, 23 Sept. 2000.


Best Student Paper Competition
Finalist
awarded to
Li Jin
For
E-Manufacturing in Networked Virtual Environments
by
Li Jin, A. Oraifige, Paul M. Lister and Frank R. Hall
presented at the:
2001 IEEE International Conference on Systems, Man and Cybernetics,
Tucson, Arizona.
October 7 - 10, 2001
Pierre E. BORNE, IEEE SMC President
Terry Bahill, 2001 General Conference Chair


WEB 3D
STUART DREDGE

References

Astheimer, P., Dai, F., Felger, W., Gobel, M., Haase, h., Muller, S., and Ziegler, R., 1995, "Virtual Design II-an advanced VR system for industrial applications," in *Proceedings of Virtual Reality World'95*, pp.337-363.

Adobe Atmosphere, http://www.adobe.com/products/ atmosphere/main.html, last accessed on 27th March, 2003.

Bailey, M. 1995. "Tele-Manufacturing: Rapid Prototyping on the Internet," *IEEE Computer Graphics and Applications*, Nov/Dec, 1995.

Baraff, D. and Witkin, A. 1997. "Physically Based Modelling: Principles and Practice," *ACM SIGGRAPH 1997 Course notes*, http://www.cs.cmu.edu/~baraff/sigcourse/index.html, last accessed on 11th July, 2002.

Barrus, J.W, Waters, R.C. and Anderson, D.B. 1996. "Locales and beacons: Efficient and Precise Support for Large Multi-user Virtual Environments." in Proceedings of the Virtual Reality Annual Symposium VRAI96. IEEE Neural networks Council.

Bauer, K. M. 1998. "Automatic Generation of Virtual Worlds for Electronic Commerce Application on the Internet," *Virtual Worlds on the Internet*, IEEE Computer Society Press, USA, pp. 207-218.

Bauer, W., Breining, R. and Roessler, A. 1995. "Co-operative Virtual Planning and Design," in *Proceedings of Virtual Reality World'95*, Stuttgart, Germany, pp.213-223.

BBC Online 2001. British Broadcasting Corporation online http://www.bbc.co.uk/history/3d.shtml, last accessed on 11th July, 2002.

Benford, S. et.al. 1995. "Networked Virtual Reality and Cooperative Work", *Presence,* MIT Press.

Benford, S., Greenhalgh, C., Rodden, T. and Pycock, J. 2001. "Collaborative Virtual Environments", *Communications of the ACM*, July 2001/ vol.44, no.7, pp.79-85.

Bennett, J. and Partridge, C. 1999. "Packet Reordering is Not Pathological Network Behaviour," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, Dec., pp.789-798.

Booch, G., Rumbaugh, J. and Jacobson, I. 1999. *The Unified Modelling Language User Guide*, Addision Wesley, ISBN 0-201-57168-4.

Borland 2003. C++ Builder Studio, http://www.borland.com/cbuilder/index.html.

Brodlie, K. et al. 2000. "Using web-based Computer Graphics to teach surgery," *Computer & Graphics*, vol. 24, pp. 157-161.

Broll, W. 1998. "DWTP-An Internet protocol for Shared Virtual Environments," in Proceedings of 3rd Symposium on the Virtual reality Modelling Language VRML98. ACM SIGGRAPH.

Brutzman, D. 1997. "Graphics Internetworking: Bottlenecks and Breakthroughs," Addison-Wesley, ACM SIGGRAPH Press.

Burdea, G. C. 1999. "Invited Review: The Synergy between Virtual Reality and Robotics," IEEE Transactions on Robotics and Automation, vol. 15, no. 3, June, pp. 400-410.

Burton, N. D., Kilgour, A. C. and Taylor, H. 1998 "A Case Study in the Use of VRML 2.0 for Marketing a Product," *Virtual Worlds on the Internet*, IEEE Computer Society Press, USA, pp. 219-236.

Capps, M. and Stotts, D. 1997. "Research Issues in Developing Networked Virtual Realities", Workshop report for "Distributed System Aspects of Sharing a Virtual Reality, *6th Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises*, Cambridge, June 18-20.

Carey, R. and Bell, G. 1997. *The Annotated VRML 2.0 Reference Manual*, Addison-Wesley, the United States of America.

Carlsson, C. and Hagsand, O. 1993. "DIVE-A Platform for Multi-user Virtual Environments," *Computer & Graphics*, 17(6).

Carson, G. S., Puk, R. F. and Carey, R. 1999. "Developing the VRML 97 International Standard", *IEEE Computer Graphics and Applications*, vol. 19, no. 2 March/April, pp. 52 - 58.

Carson, J.A. and Clark, A.F. 1999. "Multicast Shared Virtual Worlds Using VRML97," in *Proceedings of 4th Symposium on the Virtual Reality Modelling Language VRML99*. ACM SIGGRAPH.

Caudell, T. P. and Mizell, D. W. 1992. "Augmented Reality: An Application of Heads-Up Display Technology to manual Manufacturing Process," *Proceeding of 1992 IEEE Hawaii International Conference on System Science*, IEEE Press, January 1992.

Cera, C., Regli, W., Braude, I., Shapirstein, Y. and Foster, C. 2002 "A Collaborative 3D Environment for Authoring Design Semantics," *IEEE Computer Graphics and Applications*, May/June, 2002.

Chiyokura, H. 1988. *Solid Modelling with Designbase: Theory and Implementation*, Addison-Wesley Publishing.

Clark, C. and Clark, A. F. 1996. "VRML Interfaces to Information Systems" *Virtual Worlds on the Internet*, IEEE Computer Society Press, USA, pp. 21-34.

Constantine, L. and Lockwood, L. 1993. *User Interface Prototype (Software for Use: A Practical Guide to the Models and Methods of Usage-Centred Design*, (ACM Press Series)

Dai, F. and Gobel, M. 1994 "Virtual Protoyping-an Approach using VR-techniques," in *Proceedings of the 14th ASME Int. Computers in Engineering Conference*, Minneapolis, Minnesota, Sep 11-14, 1994.

Dai, F., Felger W., Fruhauf, T., Gobel, M., Reiners, D., and Zachmann, G, "Virtual prototyping examples for automotive industries," in *Proceeding of Virtual Reality World'96*, Stuttgart, 1996.

Dauner, J., Landauer, J., Stimpfig, E., and Reuter, D. 1998. "3D Product Presentation Online: The Virtual Design Exhibition," in *Proceedings of 3rd Symposium on the Virtual Reality Modelling Language VRML98.* ACM SIGGRAPH.

Deering, S. and Cheriton, D. 1990. "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. Computer Systems,* Vol. 8, No. 2, May, pp.85-110.

Deering, S. 1991. *Multicast Routing in a Data-gram Internetwork*, doctoral thesis, Stanford Univ., Dec.

Dieter W. and Jucknath, O. 1996. "MRTspace — Multi-user 3D Environments using VRML," in *Proceedings of Webnet96*, San Francisco, Association for the Advancement of Computing in Education.

DIVE- a CVE development environment, http://www.sics.se/dive, last accessed on 11th July, 2002.

EAI 2001. External Authoring Interface, Working Group. http://www.vrml.org/WorkingGroups/vrml-eai/, last accessed on 11th July, 2002.

England, D. et al. 1998. "A Virtual Environment for Collaborative Administration," *Virtual Worlds on the Internet*, IEEE Computer Society Press, USA, pp. 237-252.

Ennis, G. and Lindsay, M. 2000. "VRML Possibilities: The Evolution of the Glasgow Model," *IEEE Multimedia,* vol. 7, no. 2, April-June.

Fisher, H. 2002. "Multicast Issues for Collaborative Virtual Environments," *IEEE Computer Graphics and Applications*, September/October, pp. 68-75.

Fishwick, P.A. "3D Behavioral Model Design for Simulation and Software Engineering," in *Proceedings of 5th Symposium on the Virtual Reality Modelling Language and Web3D Technologies VRML2000*, ACM SIGGRAPH.

Flerackers, C. et al. 2001. "Creating Broadcast Interactive Drama in an NVE." *IEEE Computer and Applications,* vol. 21, no. 1, January/February, pp. 56-60.

Fuchs, H., Kedem, Z.M., and Naylor, B.F. 1980, "On Visible Surface Generation by a Priori Tree Structures." *Computer Graphics*, vol. 14, no. 3, June, pp. 124-133.

Foley, J.D., Van Dam, A., Feiner, S.K., and Hughes, J.F. 1997. *Computer Graphics: Principles and Practice (Second Edition in C)*. Addition-Wesley, United States of America.

Galitz, W.O. 2002. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, 2nd Edition,* ISBN: 0-471-08464-6, May.

Gausemeier, J., Ebbesmeyer,P. and Grafe, M., 1995. "Realistic Modelling of the Manufacturing Area of a Virtual Enterprise," in *Proceedings of Virtual Reality World'95*, Stuttgart, Germany, pp.193-203.

Ghee, S., Pausch R., and Pimentel, K., 1995. "Programming Virtual worlds," *SIGGRAPH'95* course note.

Grebner, K. and may, F. 1995. "Applications of Virtual Reality Techniques in the Industry Selected Examples," in *Proceedings of Virtual Reality World'95*, Stuttgart, Germany, pp.451-468.

Greenhalgh, C. 1996. Spatial Scope and Multicast in Large Virtual Environments, Technical report TR-96-7, Dept. of Computer Science, University of Nottingham, UK.

Greenhalgh, C., and Benford, S., 1995. "Virtual Reality Tele-conferencing: Implementation and Experience," in Proceedings of *ECSCW'95*.

Greenhalgh, C., and Benford, S. 1995. "MASSIVE: a Distributed Virtual Reality System Incorporating Spatial Trading," in *Proc. IEEE 15th International Conference on Distributed Computing Systems (DCS'95)*, Vancouver, Canada, May 30 - June 2, IEEE Computer Society.

Greenhalgh, C., and Benford, S., 1995. "MASSIVE: A Virtual Reality System for Tele-conferencing", *ACM Transactions on Computer Human Interfaces (TOCHI)*, 2 (3), pp. 239-261, ISSN 1073-0516, ACM Press, September.

Greenhalgh, C., Purbrick, J. and Snowdon, D. 2000 "Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring", Proc. CVE2000, San Francisco.

Gueziec, A., Taubin, G, horn, B., and Lazarus, F. 1999 "A Framework for Streaming Geometry in VRML," *IEEE Computer Graphics and Applications*, 29(2), IEEE Computer Society.

Hagsand, O. 1996. "Interactive Multi-user Ves in the DIVE system." IEEE Multimedia 3, 1, IEEE Computer Society. Spring, p.30-39.

Halliday, S. and Green, M. 1994. "A Geometric Modelling and Animation System for Virtual Reality". *Virtual Reality Software and Technology*, pp.71-84.

Hardwick, M., Spooner, D., Rando, T. and Morris, K. 1996. "Sharing Manufacturing Information in Virtual Enterprises," *Communication of the ACM* 39(2), pp.46-54.

Henderson-Sellers, B. and Unhelkar, B. 2000. *Open modelling with UML.* Addision-Wesley, ACM Press.

Hijiri, T., Nishitani, K., Cornish, T., Naka, T., and Asahara, S. 2000. " A Spatial hierarchical Compression Method for 3D Streaming Animation," in *Proceedings of 5th Symposium on the Virtual Reality Modelling and Web3D Technologies, VRML2000*, ACM SIGGRAPH.

Huber, A. 2001. http://www.epa.gov/gisvis/arcview/, last accessed on 11th July, 2002.

Huitema, C. 1995. *Routing in the Internet*, Prentice-Hall PTR, Upper Saddle River, N.J..

IBM Theatre Projects 2001. http://www.ibm.com/sfasp/theatre.htm, last accessed on 11th July, 2002

Jern, M. 1998. "Information Drill-Down Using Web Tools," *Virtual Worlds on the Internet*, IEEE Computer Society Press, USA, pp. 71-84.

Jin, L. and Wen, Z. 2001. "Adorning VRML Worlds with Environmental Aspects," *IEEE Computer Graphics and Applications*, vol. 21, no. 1, Jan/Feb, pp. 6-9.

John, N. W. et al. 1999. "A VRML Simulator for Ventricular Catheterisation." *Proc. Eurographics UK, Cambridge.*

Jung, B. and Milde, J.T. 1999. "An Open Virtual Environment for Autonomous Agents Using VRML and Java," in *Proceedings of 4th Symposium on the Virtual Reality Modelling Language, VRML99*. ACM SIGGRAPH.

Koenen, R. 1999. "MPEG-4: Multimedia for Our Time," *IEEE Spectrum*, February 1999, pp. 26-33.

Kraftcheck, J., Dani, T. and Gadh, R. 1997. "State of the art in virtual design and manufacturing," *VR News*, May, pp. 16-20.

Kumar, V. 1995. *MBone: Interactive Multimedia on the Internet*. New Riders, Indianapolis, Indiana,

Kreitler, M., Heim, J. and Smith, R. 1995. "Virtual Environments for Design and Analysis of Production Facilities," *IFIP WG 5.7 Working Conference on*

Managing Concurrent manufacturing to Improve Industrial Performance, Seattle, Washington, U.S.A. Sep 11-15.

Lanzagorta, M. et al. 2000. "Rapid Prototyping of Virtual Environments," *IEEE Computing in Science & Engineering,* vol. 2, no. 3, May/June, pp. 68-73.

LBNL 2000, Lawrence Berkeley National Laboratory - MBone Info.(http://www-itg.lbl.gov/ ~clarsen/vconf/vconf-faq.html)

Locke, J. 1997. "An Introduction to the Internet Networking Environment and SIMNET/DIS," http://www-npsnet.cs.nps.navy.mil/npsnet/publications/DISIntro .ps.Z

Loftin, R.B. and Kenney, P. J. 1995. "Training the Hubble Space Telescope Flight Team," *IEEE Computer Graphics and Applications*, vol. 15, no. 5, September, pp. 31-37.

Lubke, D. et. al. 2002. *Level of Detail for 3D Graphics, 1st edition.* Morgan Kaufmann.

Macedonia, M.R., Zyda, M.J., Pratt, D., Brutzman, R., Donald, P., and Barham, P.T. 1995. "Exploiting reality with multicast groups: A network architecture for large-scale virtual environments." in *Proceedings IEEE Virtual Reality Annual International Symposium* (VRAIS'95), North Carolina, March.

Manninen, T. 2000. "Rich Interaction in Networked Virtual Environments." *Multimedia: Proc. the 8th ACM international conference on multimedia*, pp. 517 - 518.

MASSIVE and inhabited television: www.crg.cs.nott.ac.uk/research/systems last accessed on 3rd Nov. 2002.

Matsuba, S. N. and Roehl, B. 1999. "Bottom, Thou Art Translated: the Making of VRML Dream," *IEEE Computer Graphics and Applications*, vol. 19, no. 2, March/April, pp. 45-51.

Maxfield, J., Fernando, T and Dew, P. 1995 "Distributed Virtual Environment for Concurrent Engineering," IEEE Annual Virtual Reality International Symposium, Triangle Park, USA, 1995, pp.162-170.

McCanne, S. 1999. "Scalable Multimedia Communication Using IP Multicast and Lightweight Sessions", *IEEE Internet Computing*, Institute of Electrical and Electronic Engineers Inc. (IEEE), Vol. 3, No. 2, March/April, pp. 33 - 45.

MetaStream.com Web Site, http://www.metastream.com, last accessed on 3rd Nov. 2002.

Microsoft Corporation - Windows Media Technologies, http://www.microsoft.com/windows/windowsmedia/en/features/roadmap.asp, last accessed on 3rd Nov. 2002.

Microsoft Research - Mbone Info for Windows 95, 98 & NT, http://www.research.microsoft.com/research/BARC/mbone/, last accessed on 3rd Nov. 2002.

Mitchell, W. L. 1998. "Moving the Museum onto the Internet: The Use of Virtual Environments in Education about Ancient Egypt" *Virtual Worlds on the Internet*, IEEE Computer Society Press, USA, pp. 263-278.

Moller, T. and Haines, E. 1999. *Real-Time Rendering*, A K Peter, Ltd. Massachusetts, USA.

Mourant, R. R. et al. 1995. "A Virtual Environment for Training Overhead Crane Operators," *The transportation Research Board Annual Meeting*, Washington, DC. Jan. 22.

Microsoft Messenger, MSN, 2002. http://www.msn.com, last accessed on 5th Dec. 2002

Naka, T., Mochizuki, Y., Hijiri, T., Cornish, T., and Asahara, S. 1999. "A Compression/ Decompression Method for Streaming-Based Humanoid Animation," in *Proceedings of 4th Symposium on the Virtual Reality Modelling Language VRML99*, ACM SIGGRAPH.

NIST 2001. "Translation of Deneb Robotics Formats to VRML," Visualisation and Virtual Reality Group, National Institute Standard Technology, http://ovrt.nist.gov/projects/mfg/SIMA/deneb2vrml/deneb2vrml.html, last accessed on 11th July, 2002.

NPSNET networking 2002. http://www.movesinstitute.org/~npsnet/v/networking, last accessed on 11th July, 2002.

Payne, J.E. and Mills, R. 1992. "A Distributed Simulation Environment for Simulating Manufacturing Processes," in *Proceedings of the Computer Simulation Conference*, San Diego, pp.1169-1173.

Palmer, I. J. and Reeve, C. M. 1998. "Collaborative Theatre Set Design across Networks," *Virtual Worlds on the Internet*, IEEE Computer Society Press, USA, pp. 253-262.

Perlman, R. 2000. *Interconnections*, 2nd ed., Addison-Wesley, Boston, Mass..

Pesce, M. 1995. *Browsing and Building Cyberspace*, New Riders Publishing, Indianapolis, USA.

Planet com. 2001. http://www.planet9.com/earth/sf/, last accessed on 11th July, 2002.

PTC 2003. The Product Development Company: http://www.ptc.com/, last accessed on 6th August, 2003.

QUEST User Manuals 1995. Version 2.1, Delmia Corp., MI.

RealNetworks Inc. - Streaming Media Web Site: http://www.realnetworks.com/, last accessed on 3rd Nov. 2002.

Reddy, M. et al.1999. 'TerraVision II: Visualizing Massive Terrain Databases in VRML," *IEEE Computer Graphics and Applications,* vol. 19, no. 2, March/April.
Regli, W.C. 1997. "Internet-enabled Computer Aided Design," *IEEE Internet Computing*, Jan/Feb, pp.39-50.

Ressler, S. 1994. "Applying Virtual Environments to Manufacturing," *National Institute of Standards and Technology,* USA. http://ovrt.nist.gov/projects/mfg/mfgVRcases.html

Rhyne, T., Brutzman, D. and Macedonia, M. 1997. "Internetworked Graphics and the Web," *IEEE Computer*, vol. 30, no. 8, August, pp. 99 - 101.

Rhyne, T.M. 1999. "Internetworked 3-D Computer Graphics: Overcoming Bottlenecks and Supporting Collaboration," *ACM SIGGRAPH.*

Rolland, J., Davis, L., Ha, Y., Meyer, C., Shaoulov, V., Akcay, A., Zheng, H., Banks, R., and Vento, B. 2002 "3D Visualization and Imaging in Distributed Collaborative Environments", *IEEE Computer Graphics and Applications,* Jan/Feb, 2002.

Russ, K. and Wetherelt, A. 1999. "Large-scale Mine Visualisation Using VRML," *IEEE Computer Graphics and Applications,* vol. 19, no. 2, March/April.

Seiler, C. and Schafer, A. 1998. "MUSyC: Scalable Multi-user Virtual Environments based on VRML," in *Proceedings of Virtual Environment 98 Conference and 4th Eurographics Workshop*, Stuttgart, Germany, Eurographics.

Schulz, M., Reuding, T. and Ertl, T. 1998. "Analyzing Engineering Simulations in a Virtual Environment," *IEEE Computer Graphics and Applications*, vol. 18, no. 6, Nov./Dec., pp. 46-52.

Shyamsundar, N. and Gadh, R. 2001 . "Internet-based Collaborative Product Design with Assembly Features and Virtual Design Spaces," *Computer-Aided Design*, 33(2000), pp. 637-651.

Singhal, S. and Zyda, 1999. M. *Networked Virtual Environment: Design and Implementation,* Addision-Wesley, New York.

Smith, R. P. and Heim, J. A. 1999. "Virtual Reality Layout Design: the Value of an Interactive three-dimensional Representation," *International Journal of Production Research*, vol. 37. No. 17, pp. 3941-3957.

SPLINE, 2002. 2-An environment for online communities, http://www.merl.com/projects/ spline/index.html, last accessed on 11th July, 2002.

Stallings, W. 1996. *Data and Computer Communications*, 5th ed. Upper Saddle River, NJ: Prentice-Hall.

Stevens, W.R. 1994. *TCP/IP Illustrated, Volume 1: The Protocols*, Addision-Wesley, Boston, Mass.

Stevens, W.R. 1998.*UnixNetwork Programming Volume 1*, Prentice Hall PTR, Upper Saddle River, N.J.

Stucki, P., Bresenham, J and Earnshaw, R. 1995. "Computer Graphics in Rapid Prototyping Technology," *IEEE Computer Graphics and Applications*, Nov/Dec 1995.

Sweet, S. 2001. "A Survey of Small and Medium-sized Enterprises (SMEs) in Midlands, UK." Internal document, School of Engineering and the Built Environment, University of Wolverhampton.

Thibault, W.C. and Naylor, B.F. 1987, "Set Operations on Polyhedra Using Binary Space Partition Trees." *Computer Graphics*, vol. 21, no. 4, July, pp. 153-162.

TightVNC 2003. http://www.tightvnc.com/download.html, last accessed on 18th June, 2003.

Ultima Online 2001. http://www.origin.ea.com/, last accessed on 11th July, 2002.

UVa User Interface Group 1995. "Alice: Rapid Prototyping for Virtual Reality," *IEEE Computer Graphics and Applications*, May 1995.

Vince, J. 1995. *Virtual Reality System,* Addison-Wesley Publishing Company, New York.

Vince, J. 2001. *Essential Mathematics for Computer Graphics Fast*, Springer-Verlag, UK.

Virtual Technologies Inc. (Vti), 2001. http://www.virtex.com/products/hw_products/cybertouch.html, last accessed on 11th July, 2002.

Waters et. al. 1996. "Diamond Park and SPLINE: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability," TR-96-02a

Technical report January, Cambridge, Mass. http://www.merl.com/papers/TR96-02a/

Watt, A. 2000. *3D Computer Graphics (Third Edition)*, Addison-Wesley Publishing Company, USA.

Weinschenk, S., Jamar, P., and Yeo, S.C. 1997. *GUI Design Essentials*. ISBN: 0-471-17549-8, March 1997.

Wilson, J. et al. 1995. "Manufacturing Operations in Virtual Environments (MOVE)," *Presence----Teleop. Virtual Environ.*, vol. 4, no. 3, pp. 306-317.

Wilson, J. 2000. *AutoCAD 2000: 3D modelling: a visual approach*. Albany, NY.

Ye, N. and Dech, F. 1999. "Assembly Planning Effectiveness Using Virtual Reality," *Presence*, vol. 8, no. 2, April, pp 204-217.

Yoshiaki, A. 1998. "VSPLUS: A High-level Multi-user Extension Library for Interactive VRML Worlds," in *Proceeding of 3rd Symposium on the Virtual reality Modelling Language VRML98*. ACM SIGGRAPH.

Zhang, Y., Zhang, C. and Wang, H.P. 2000 "An Internet based STEP data exchange framework for Virtual Enterprise," *Computers in Industry*, 41(2000), 51-63.

Appendix A

The VFLS Software Toolkit User Manual and Tutorial

1. Software Installation Instructions:

The section presents how to install the software toolkit — Distributed VRML-based Factory Layout Simulator (VFLS version 1.0, developed by Li Jin).

- Go to the folder —"VFLInstall" in the attached CD, double click on "setup.exe", the installation program will come up, the follow the instructions to finish the software toolkit installation.
- The software toolkit Demo —"VFLSDemo.avi" is included in the CD. It shows how to use the toolkit.

2. The Toolkit User Manual

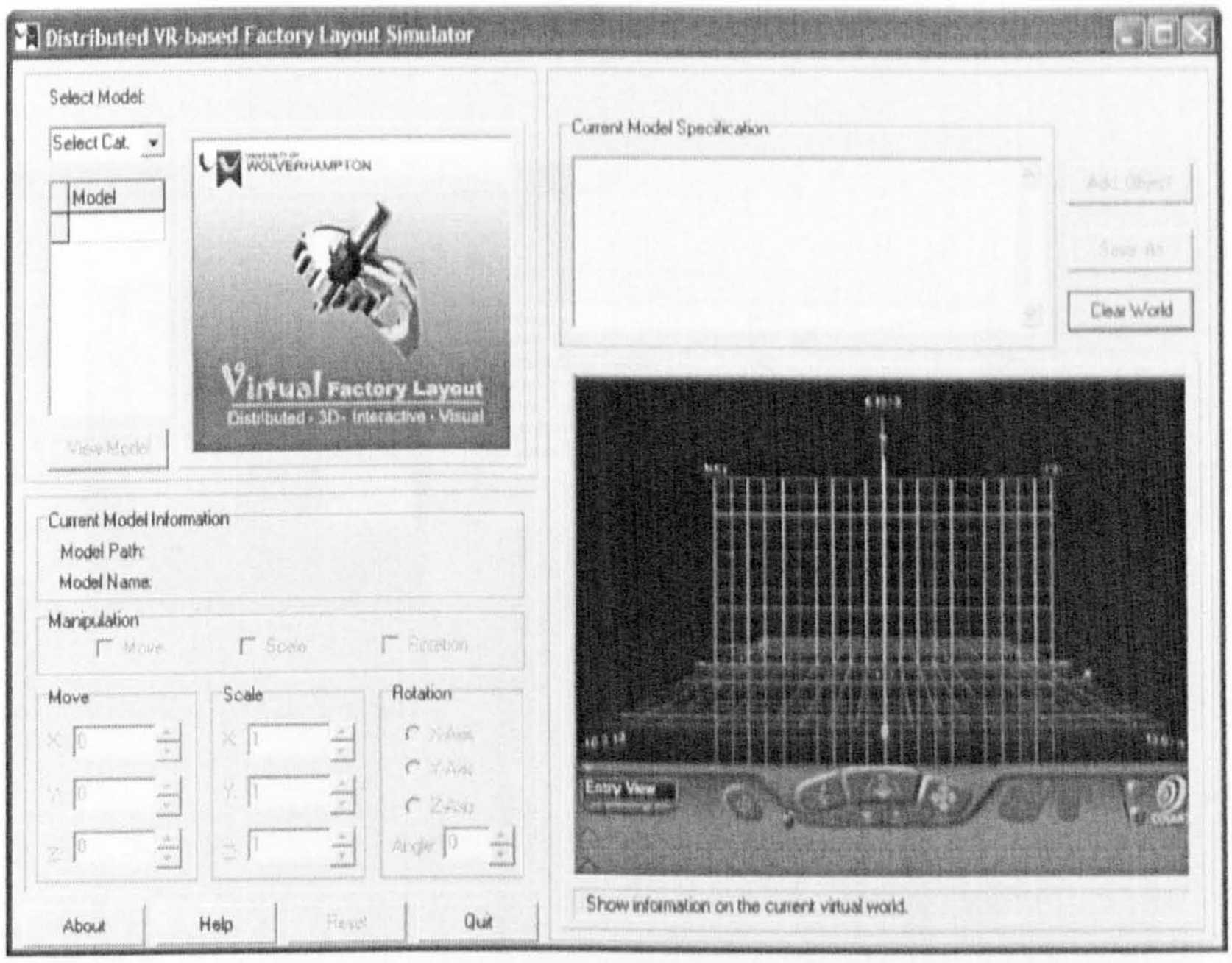
The user manual presents how to use the software toolkit — Distributed VRML-based Factory Layout Simulator (VFLS version 1.0, developed by Li Jin).

I. Starting and Quitting the VFLS Toolkit

This section shows you how to start, quit, and get help for the software Toolkit —Distributed VRML-based Factory Layout Simulator.

► To start using the toolkit

1. Click the "**Start**" button to display the **Start** menu.
2. From the "**Start**" menu, you can start the authoring toolkit by using **Programs →VICTEC → factory**.
3. You can see the Graphic User Interface (GUI) of the toolkit as follows.



► **To quit the software toolkit**

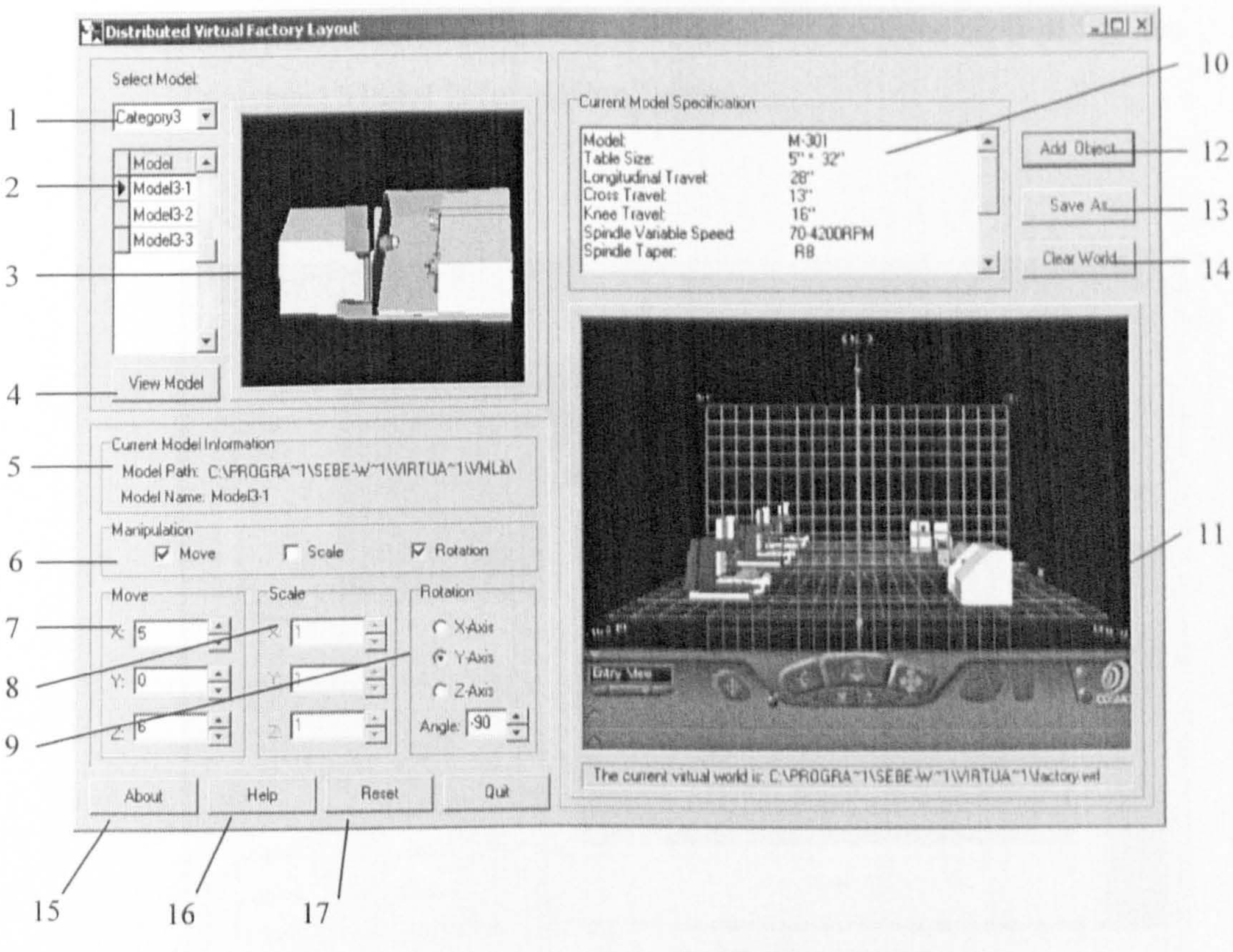
4. Click the "**Quit**" button or the ☒ to quit the software toolkit.

II. Using the VFLS Toolkit

This section shows you how to use the VFLS toolkit to quickly produce factory layouts in virtual environments by manipulating 3D models of machines selected from Virtual Machine Database (VMD).

It includes:

- How to select a 3D model of machine from the VMD;
- How to view and manipulate the selected machine in 3D manner;
- How to layout the selected machine in virtual scenes by 3D manipulation;
- How to add the selected machine in the Web-based virtual environment;
- How to save the current factory layout scene and reset the virtual environment;



► **To Select a 3D model of machine from the VMD**

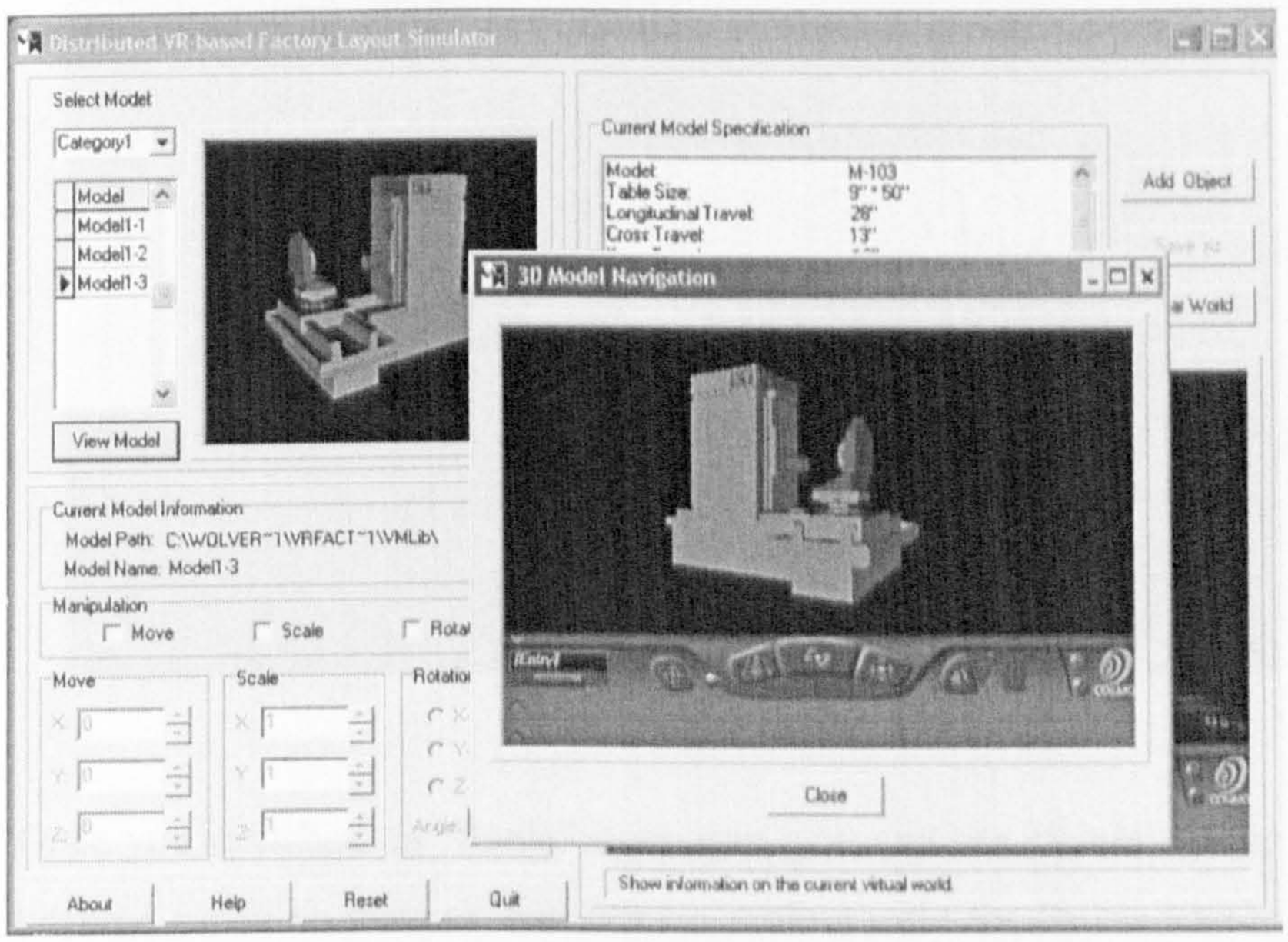
1. Click the **arrow** button beside the "Select Cat." box to show the **pull down list box** which contains the list of different categories of machines existing in VMD.
2. The list of filenames of machines belonged to the selected category appears in the list box. Scroll through the machine list and click the one that you want to select.
3. The picture of the selected machine appears in the image box.

► **To View and Manipulate the selected machine in 3D manner**

4. Click the "**View Model**" button to pop up the "**3D Model Navigation**" window as below. In this window, you are allowed exploring and viewing the selected machine's 3D model in real-time. The embedded VRML browser is able to navigate the selected

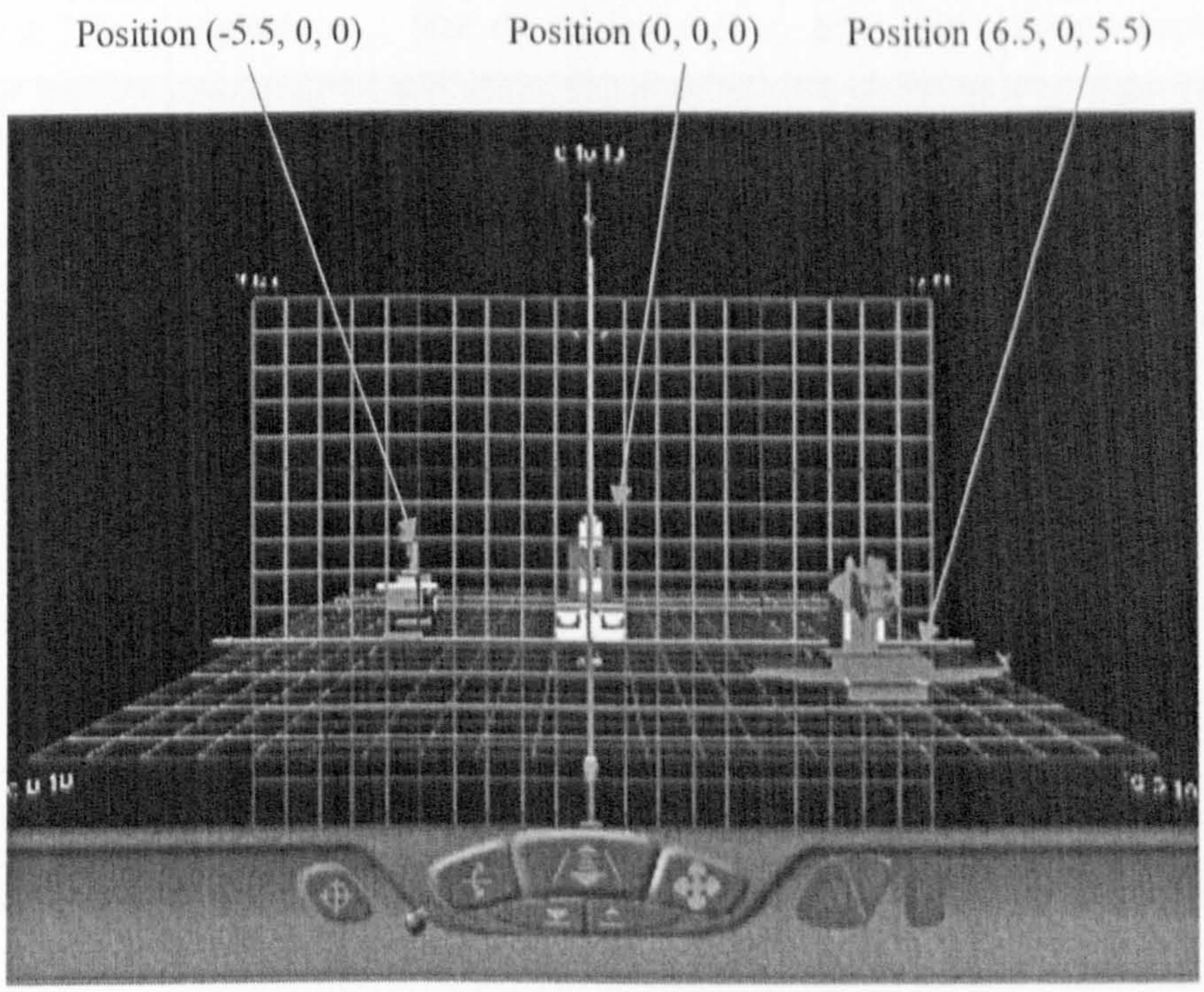
model from any viewpoints and manipulate the 3D model through rotate, pan, zoom etc.

5. The path and filename of the selected machine appear in the "**Current Model Information**" panel.

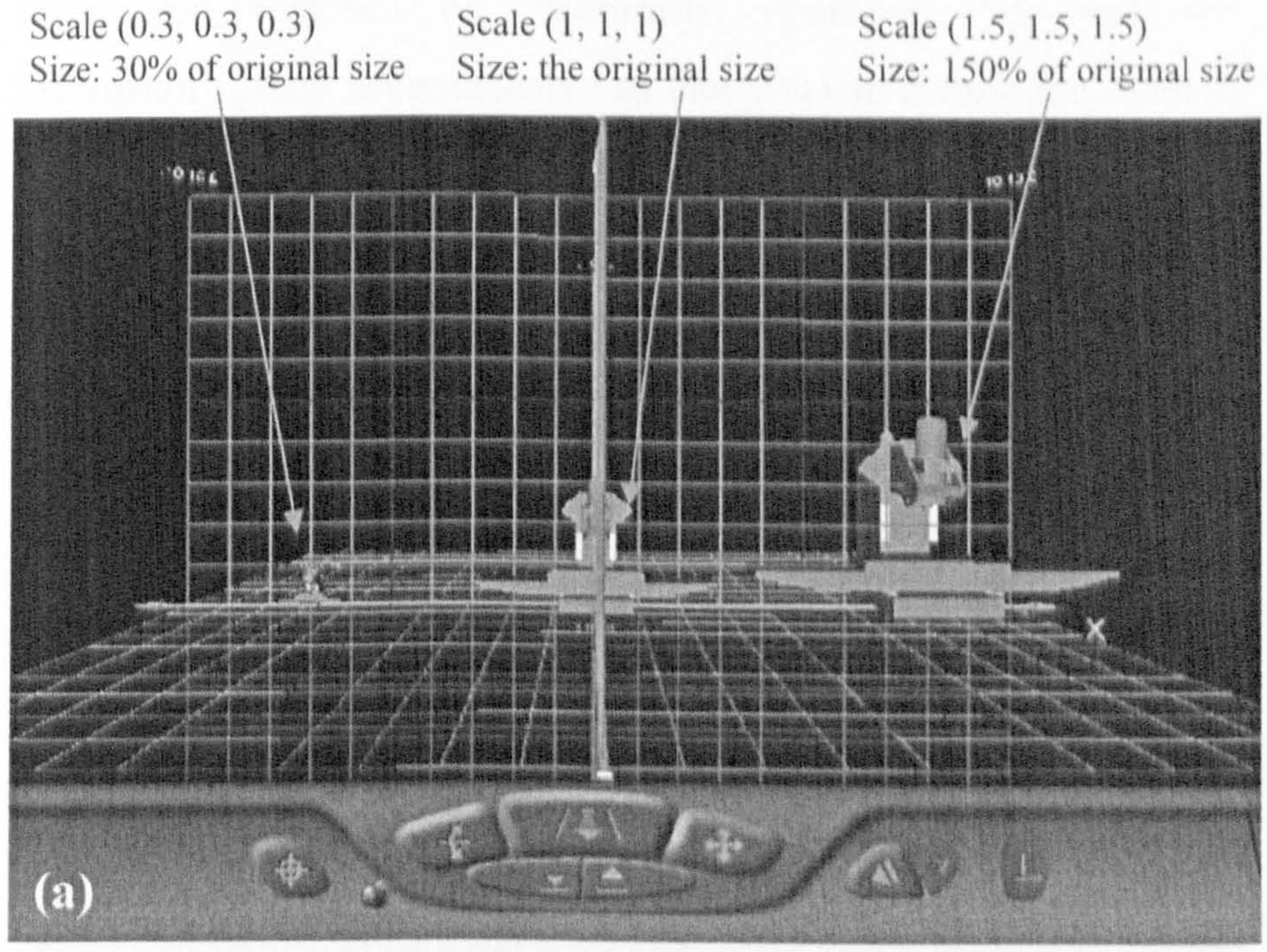


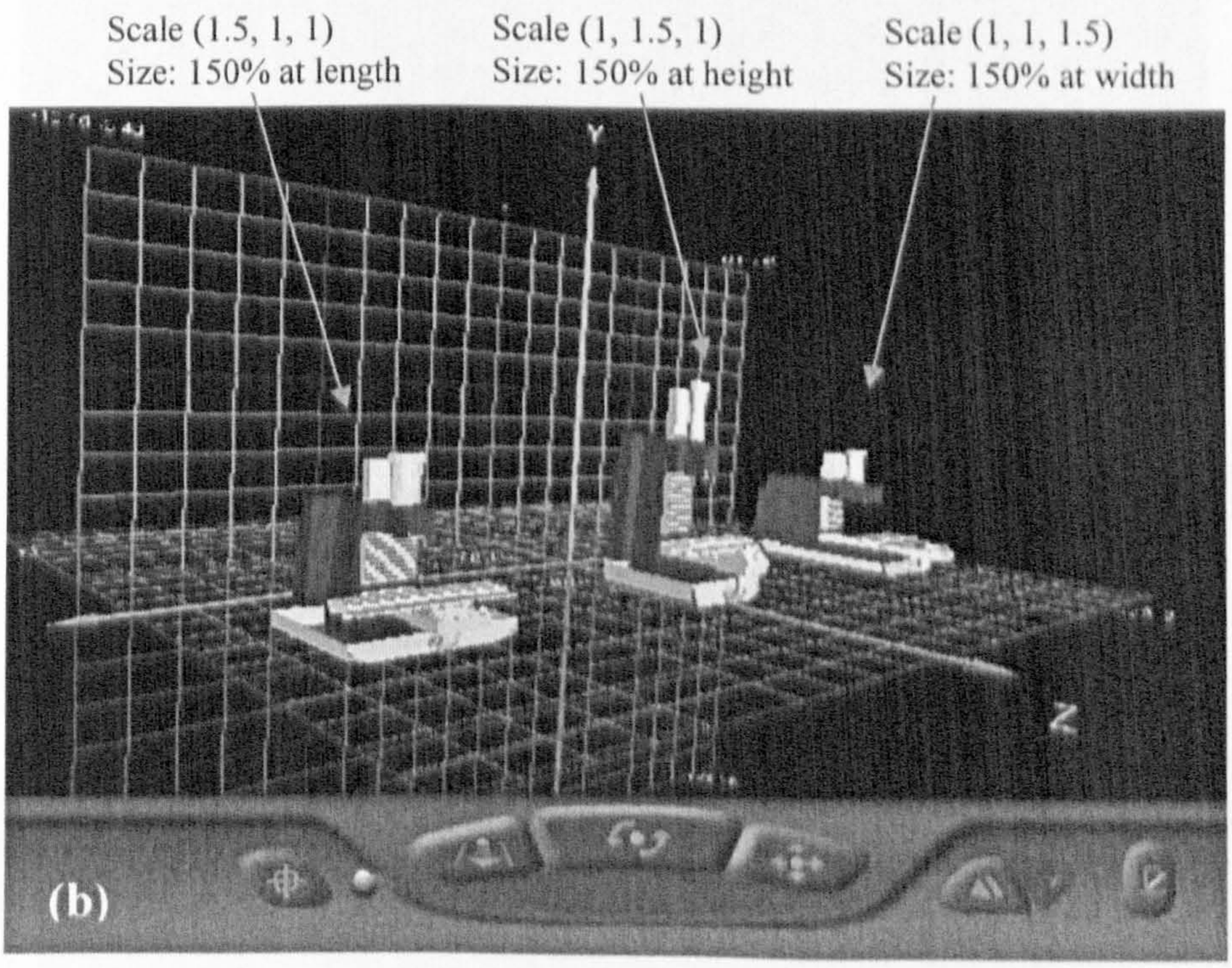
► **To Layout the selected machine in virtual scenes by 3D manipulation**

6. In the "**Manipulation" panel**, you are able to choose and constraint 3D manipulations for layout operations by switching on/off **Move**, **Scale**, and **Rotation**.
7. Once you switched on "**Move**" operation, you can see the "**Move**" group become active so that you can input an initial position value (x, y, z) to locate the selected machine in a virtual scene. For example as shown below, three machines are positioned in the virtual environment. Their initial position values input from the user are (-5.5, 0, 0) for Model1-1, (0, 0, 0) for Model2-1, and (6.5, 0, 5.5) for Model2-2.



8. Once you switched on "**Scale**" operation, you can see "**Scale**" group become active so that you can input a scaling ratio Sx, Sy, and Sz to make the selected machine scale along the respective coordinate axis x, y, z. The scale ratio should be larger than zero. If x=1, y=1, z=1, the object is the original size. If x=0.3, y=0.3, z=0.3, the object is the 30% of its original size. If x=1.5, y=1.5, z=1.5, the object is the 150% of its original size. The results of different input value for uniform scaling are shown in the following figure (a). In addition, it supports the non-uniform scaling as well. For example as shown in the following figure (b), if the scale input values are (1.5, 1, 1), the object only enlarges 50% along its X-axis as its length, and its height and width are kept as its original size.


Scale (0.3, 0.3, 0.3)
Size: 30% of original size
Scale (1, 1, 1)
Size: the original size
Scale (1.5, 1.5, 1.5)
Size: 150% of original size
(a)

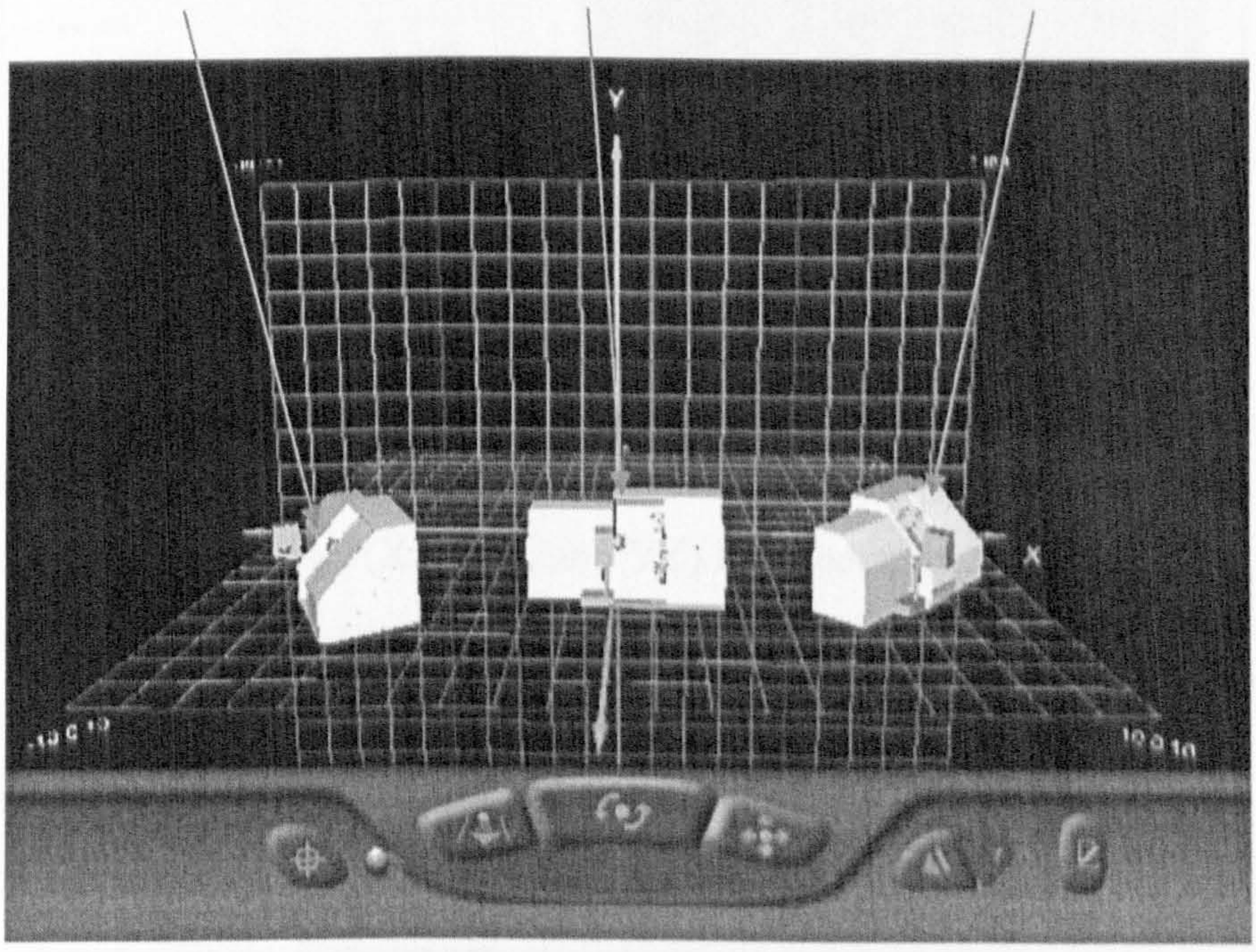

Scale (1.5, 1, 1)
Size: 150% at length
Scale (1, 1.5, 1)
Size: 150% at height
Scale (1, 1, 1.5)
Size: 150% at width
(b)

9. Once you switched on "**Rotation**" operation, you can see "**Rotation**" group become active so that you can choose the rotation axis and input the rotation angle to make the selected machine rotation around the selected axis and angle. An example shown in the following figure illustrates the results by rotating the same machine model with different angles around Y-axis. Of course, the function supports a 3D object to rotate around X-axis and Z-axis as well. However, this situation is unlikely happened to manipulate machines in factory layout application in reality.

Rotation angle: -60°
Rotation axis: Y

Rotation angle: 0°
Rotation axis: Y

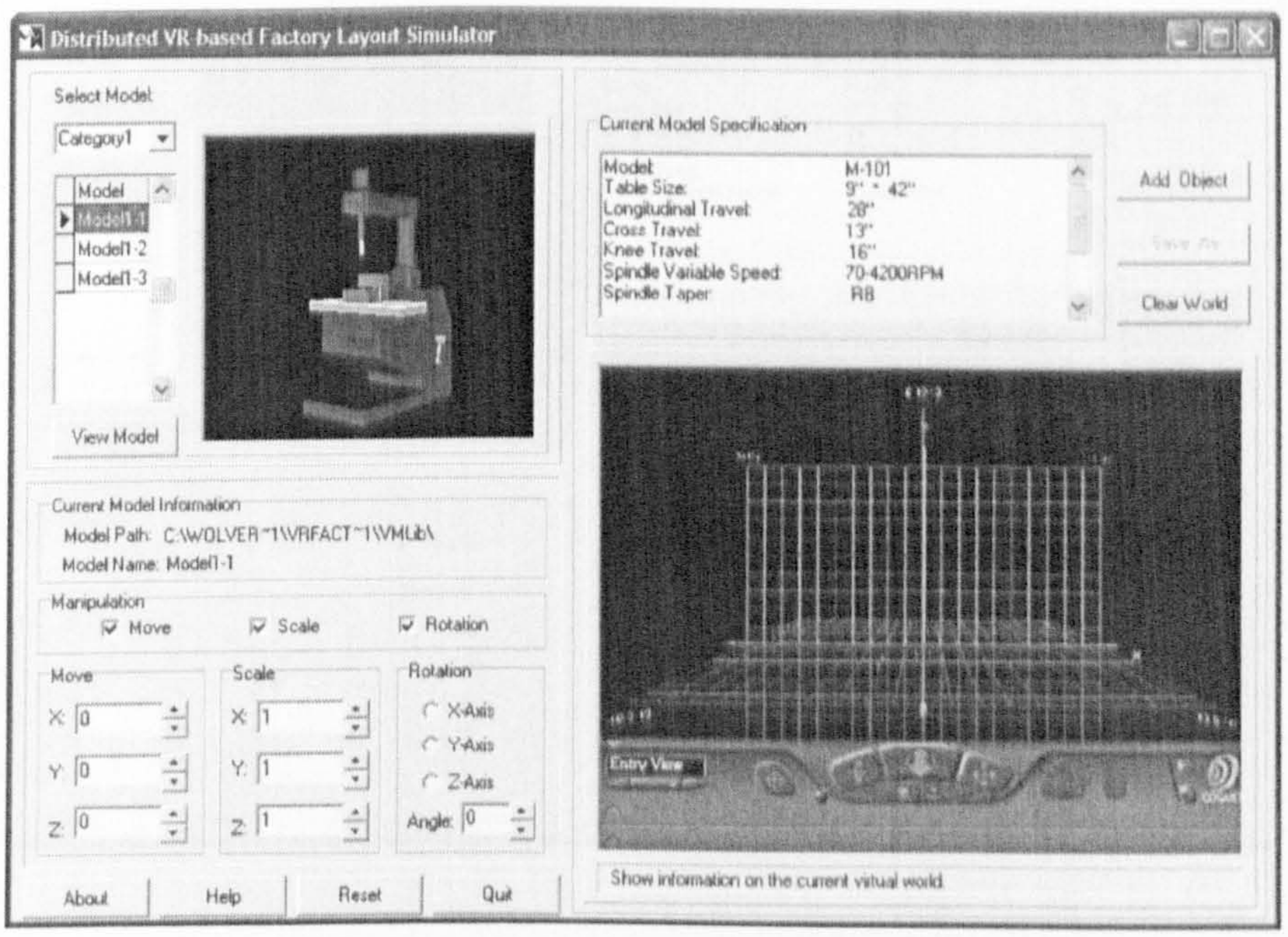
Rotation angle: 45°
Rotation axis: Y



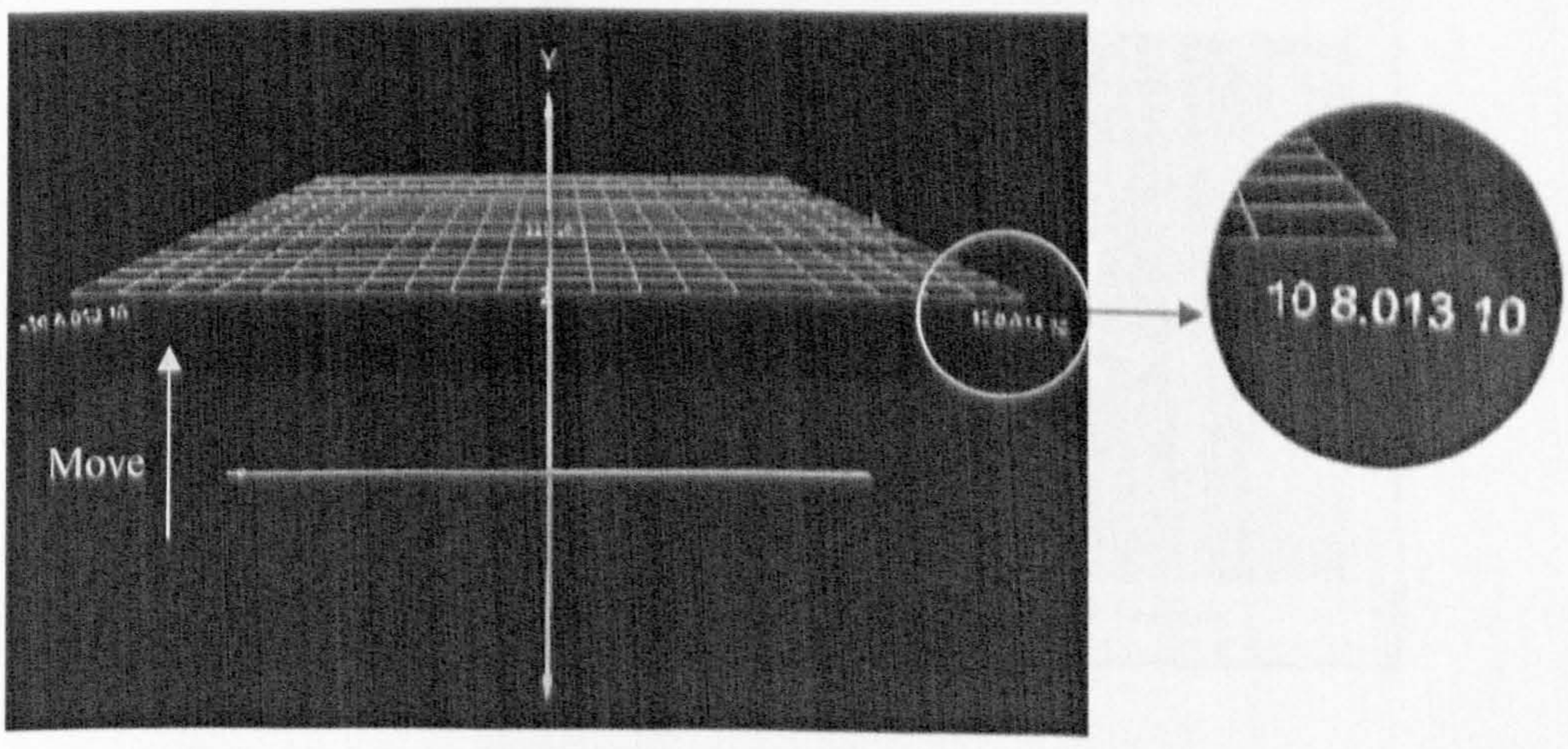
► **To Add the selected machine in a virtual environment**

10. Once you selected a machine from a category, the relevant engineering information (e.g. the machine's specification) can be extracted from the VMD and then you can read this information in the "**Current Model Specification**" box.
11. The Web browser — Internet Explorer and the CosmoPlayer plug-in

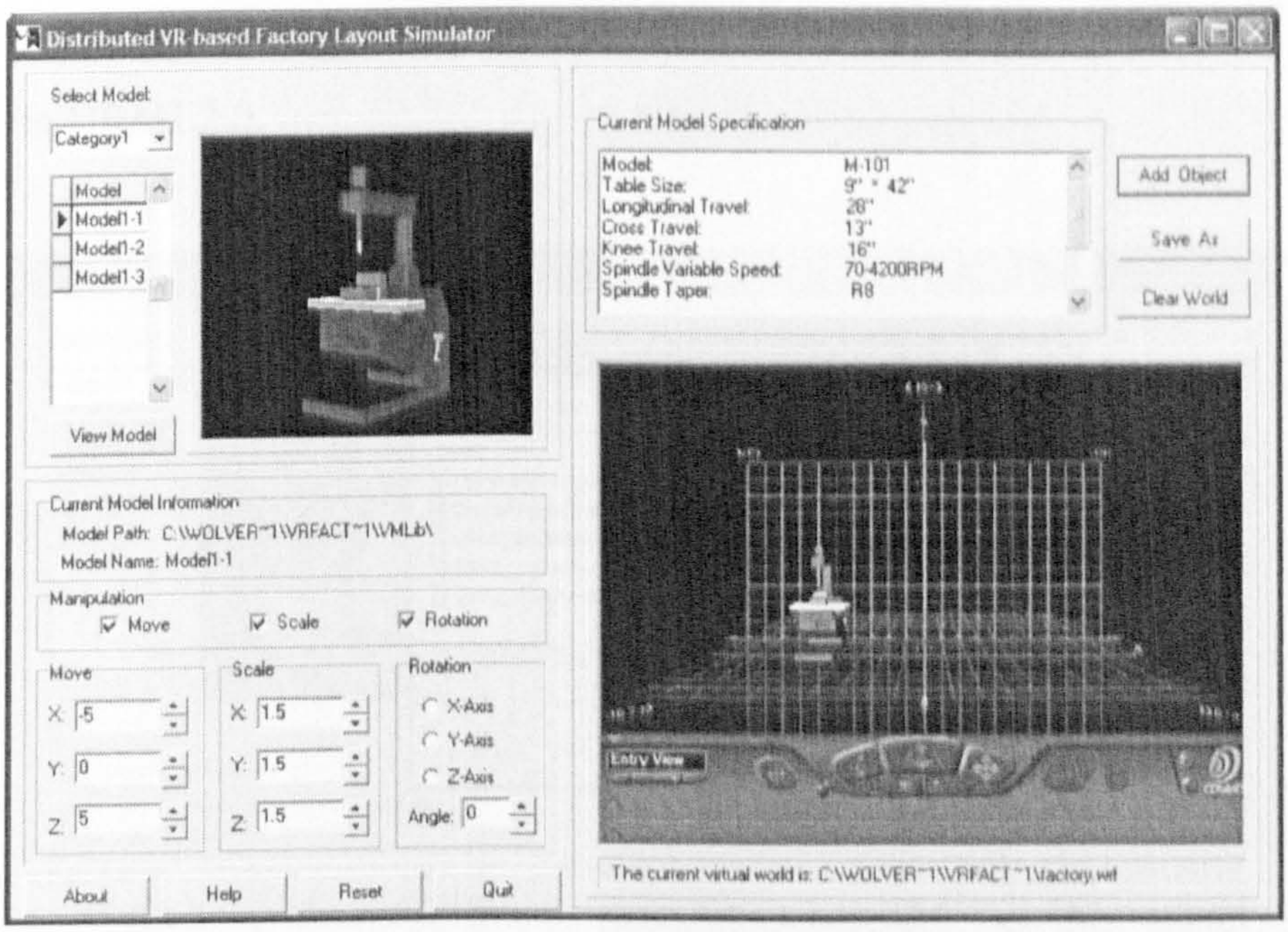
have been embedded to establish the Web-based virtual environment in this software toolkit. You can see a Web-based virtual environment which is represented by a grid-enhanced 3D coordinate system as shown in the following figure.



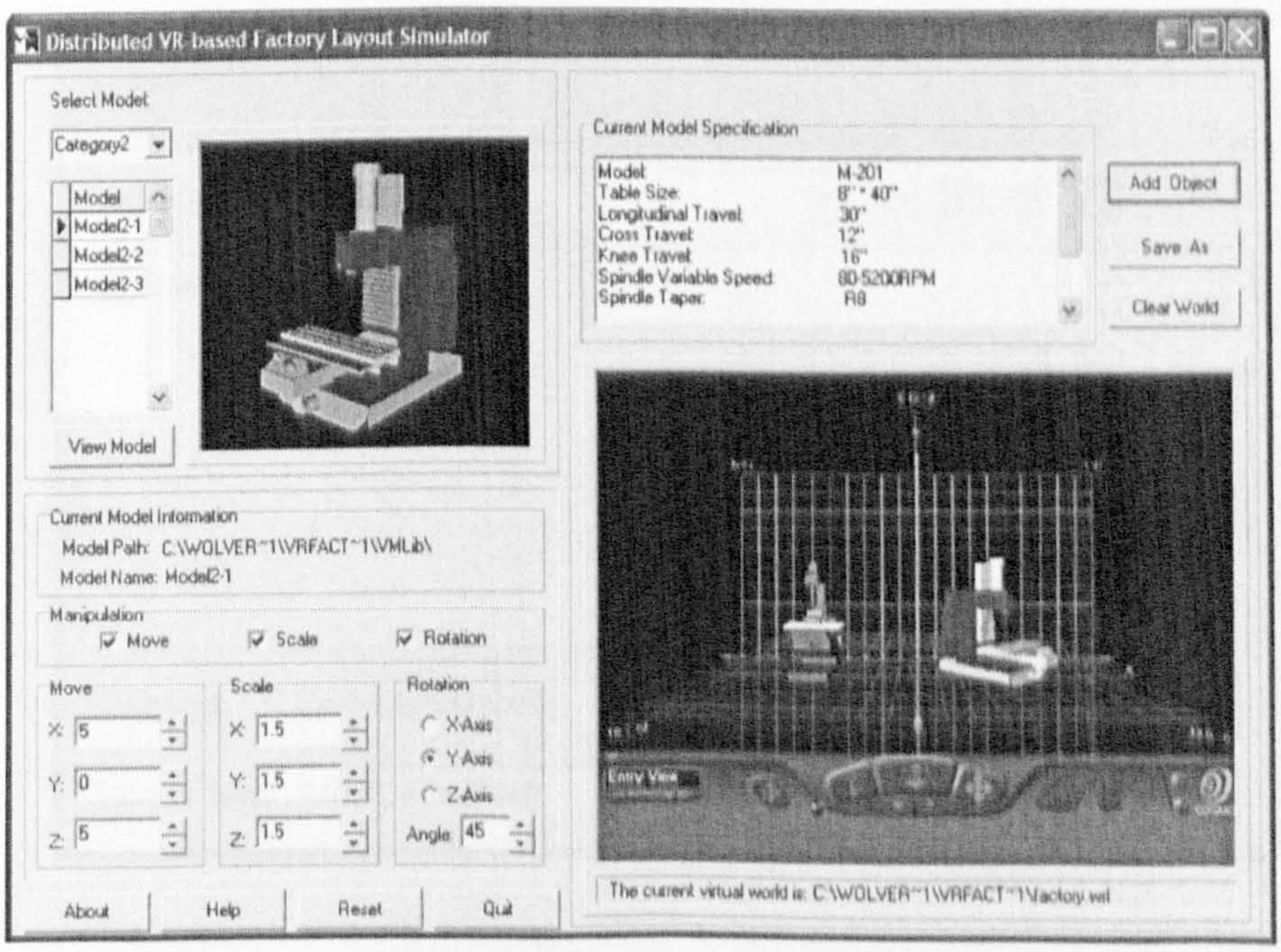
In order to enhance the ability of precise positioning in real time, the three "***Grid-planes***" (XY, YZ, and YZ) have been implemented to be able to move along the relevant axis. For an example shown in the following figure, the plane of XZ-based Grid can move along the Y-axis. With moving up and down, the precise values of points are changed and are displayed in real time within the 3D scene.



12. Click on the "**Add Object**" button to add the selected machine into the Web-based virtual environment (see 11). An example is shown as below.

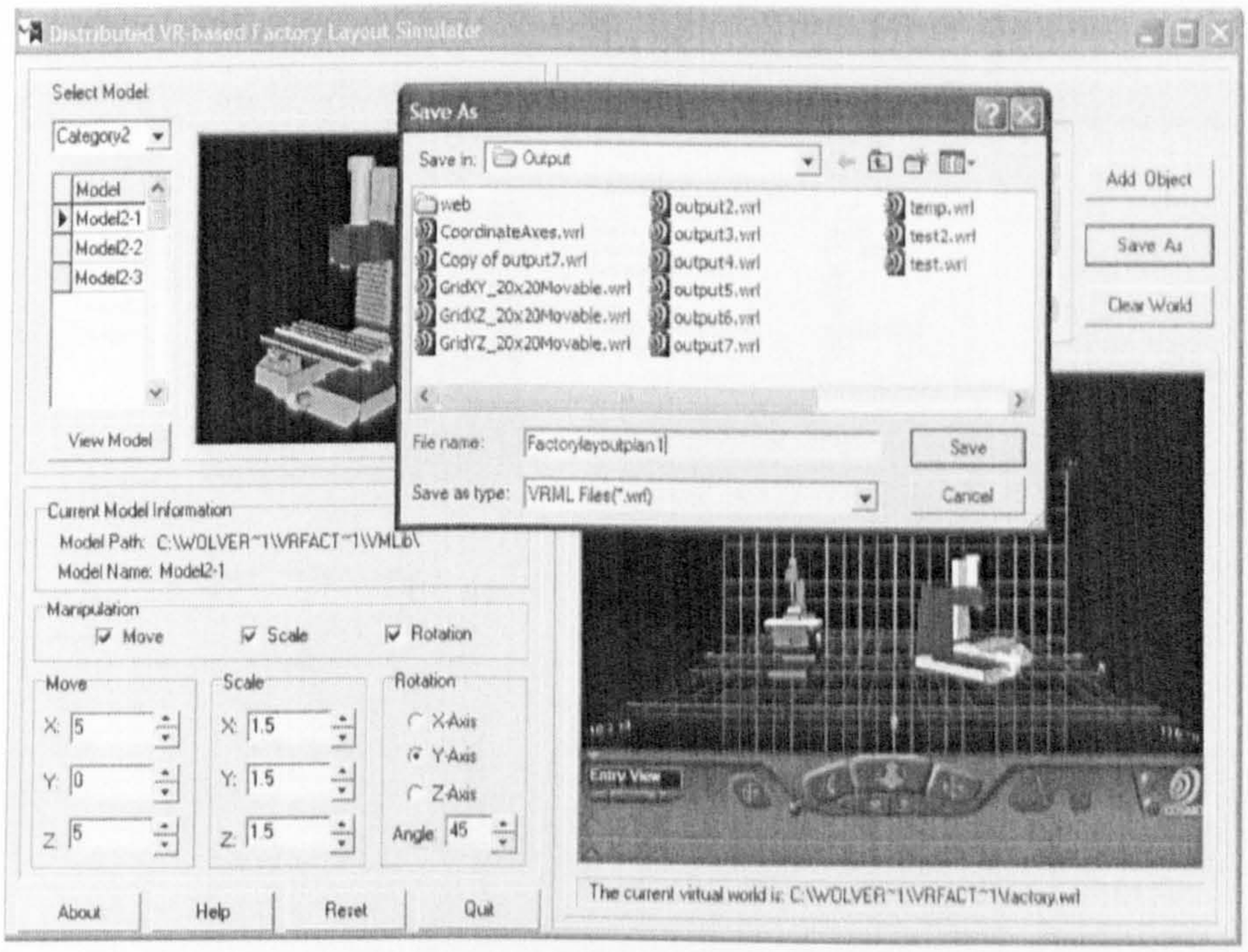


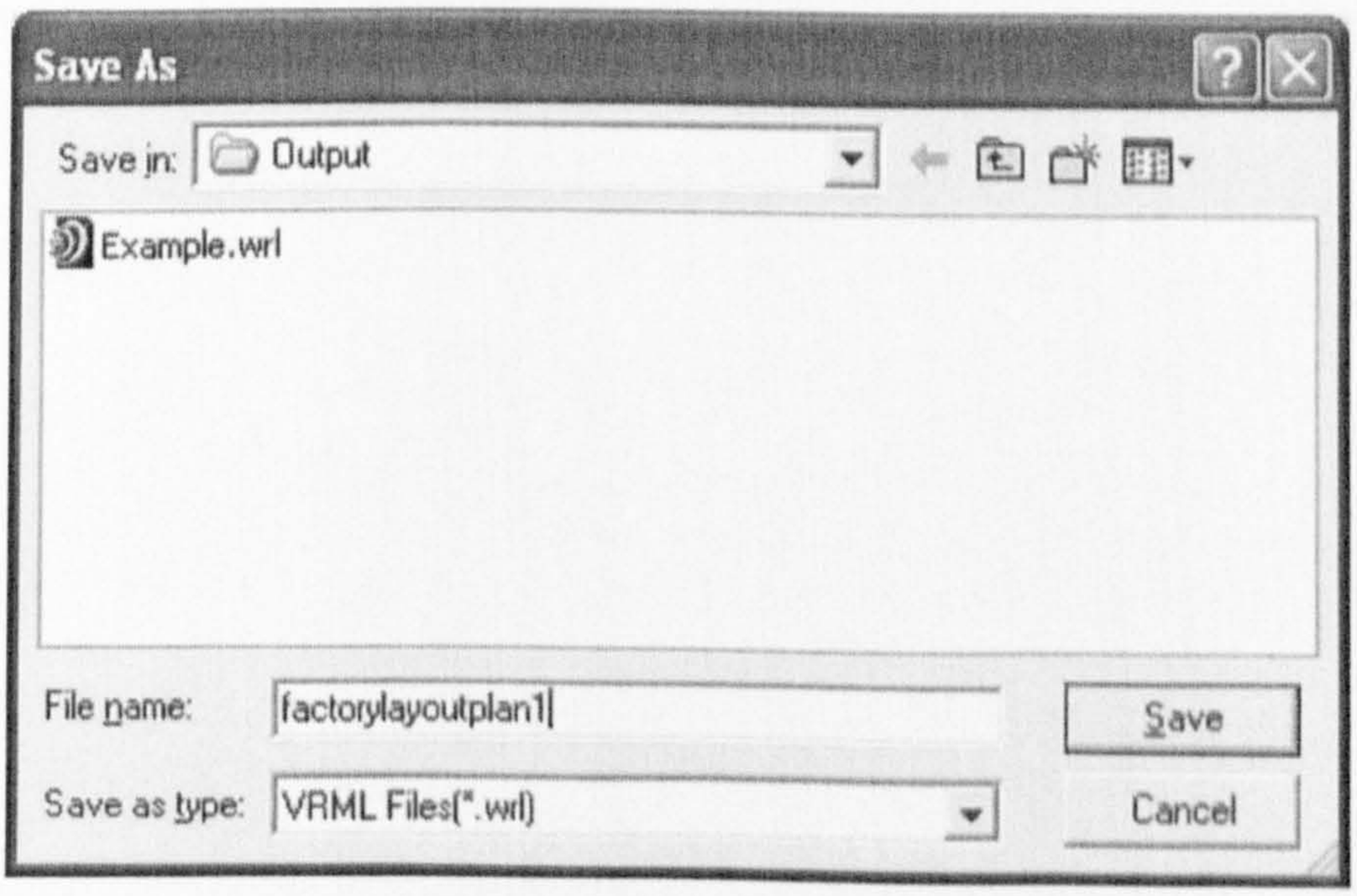
To add more machines into the same virtual scene, you need to select other machines and click on the "**Add Object**" button to add it into the current virtual environment as below.



► **To Save the current factory layout scene**

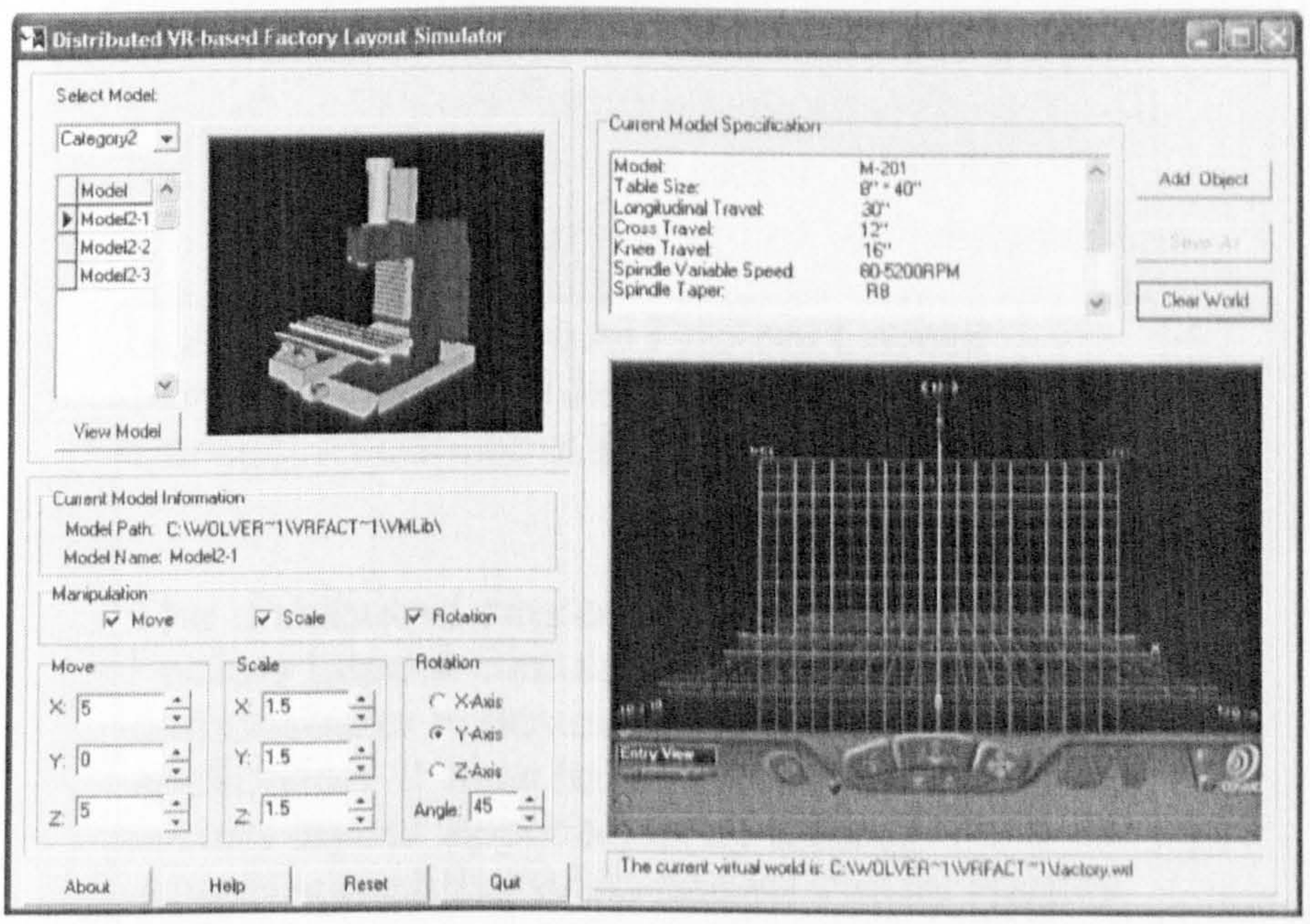
13. After finish a factory layout scene, click the "**Save As**" button to save and output the current layout scene into a VRML file. You can see the following dialog comes up allows saving your layout scene in VRML2.0.





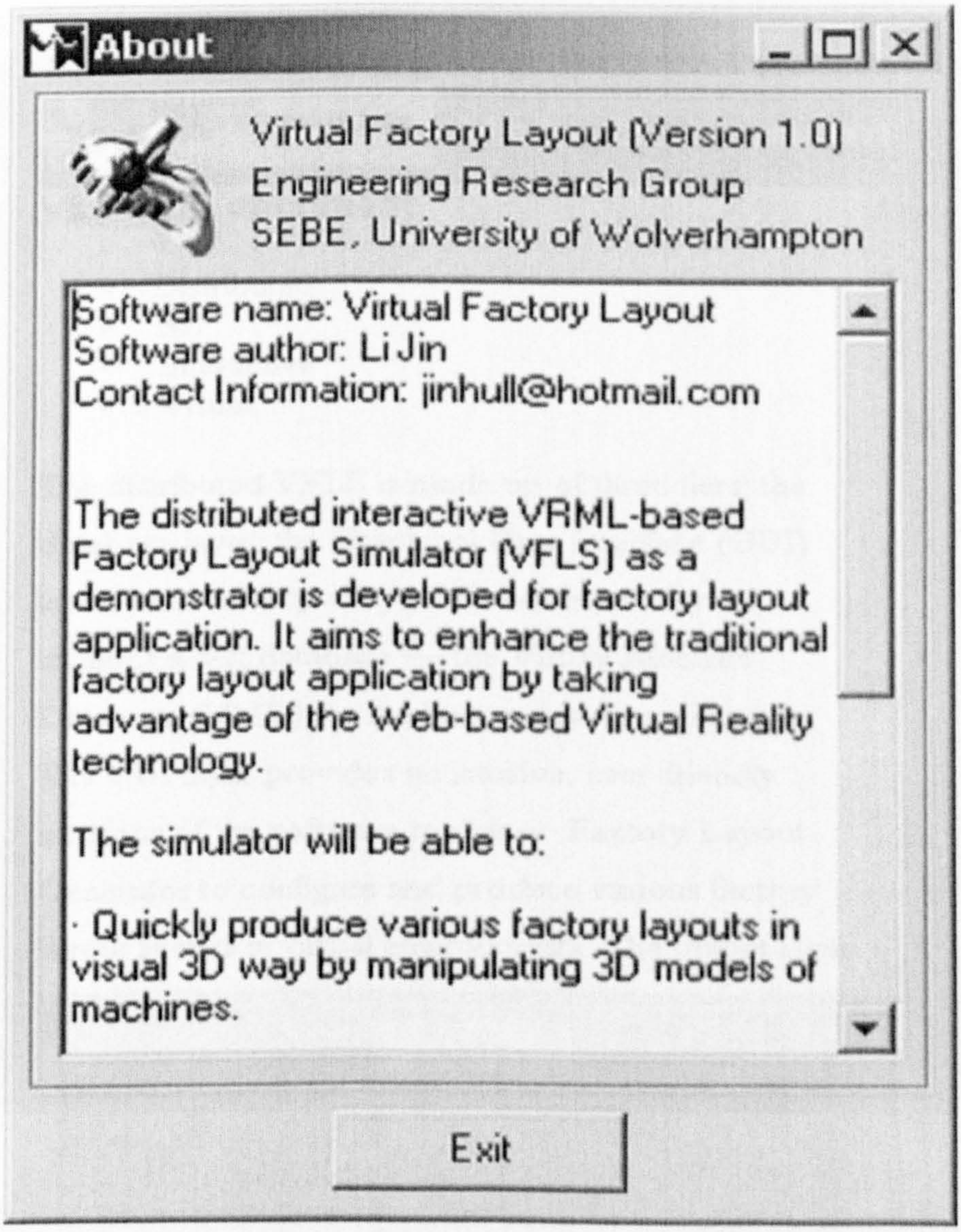
► **To Clear the current factory layout scene and reset the Web-based virtual environment**

14. After save a factory layout scene, you may want to product another factory layout, click the "**Clear World"** button to clear the current layout scene and reset the original 3D Gird-enhanced coordinate system in the Web-based virtual environment as below.



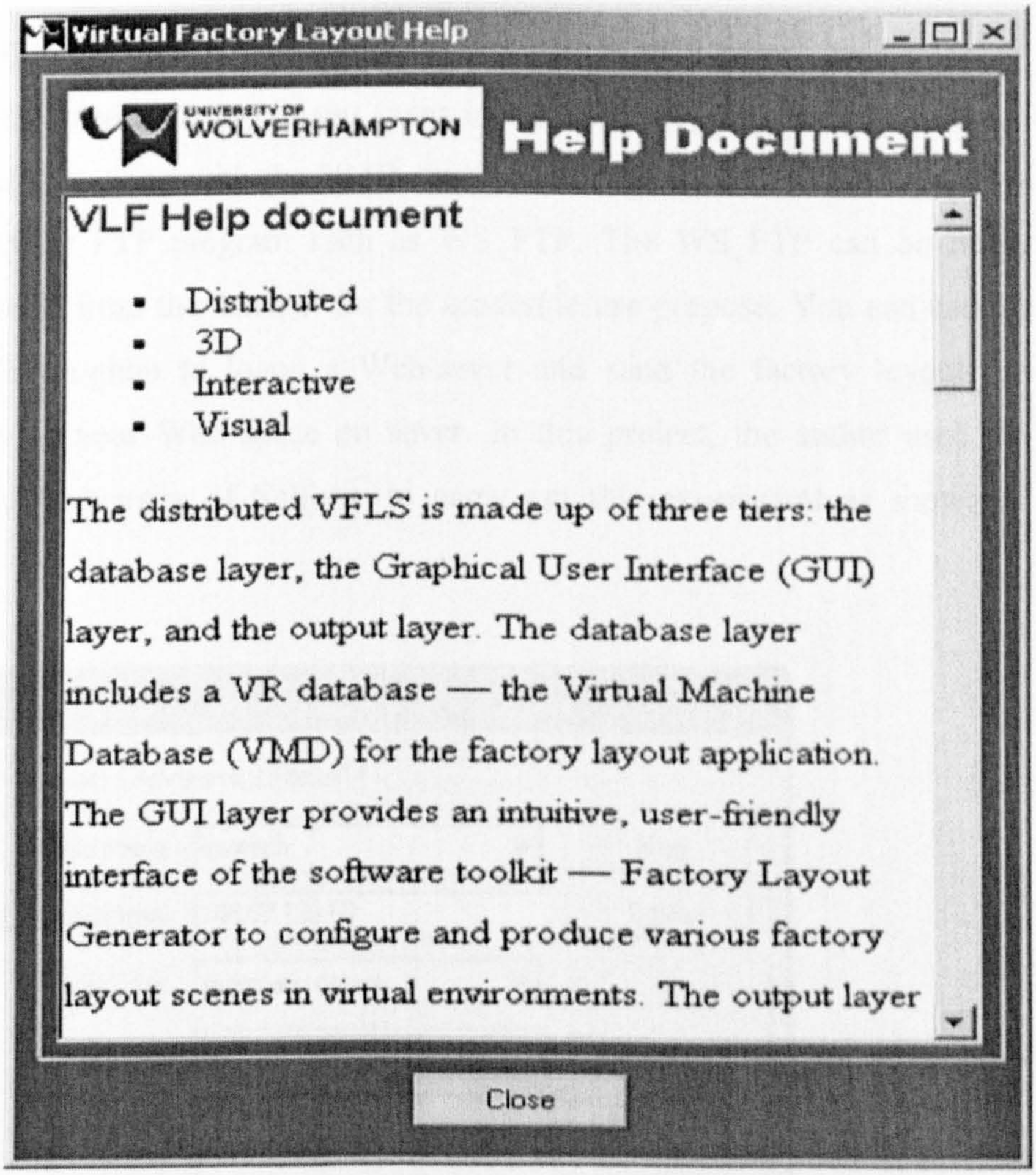
► **To get information on the current version**

15. Click the "**About**" button to show the "**About**" window as below. In this window, you can get the basic information on the current version of the software toolkit.



► **To get Help about the VFLS toolkit**

16. Click the "**Help**" button to show the "**Help**" window as below. You can see the help document about the software toolkit — Distributed VRML-based Factory Layout Simulator.



► **To Reset the VFLS toolkit**

17. Click the "**Reset**" button to reset all input values from the user to system default: position (0, 0, 0) for move group, the original size of object (1, 1, 1) for the scale group, and Y-axis for rotation axis and

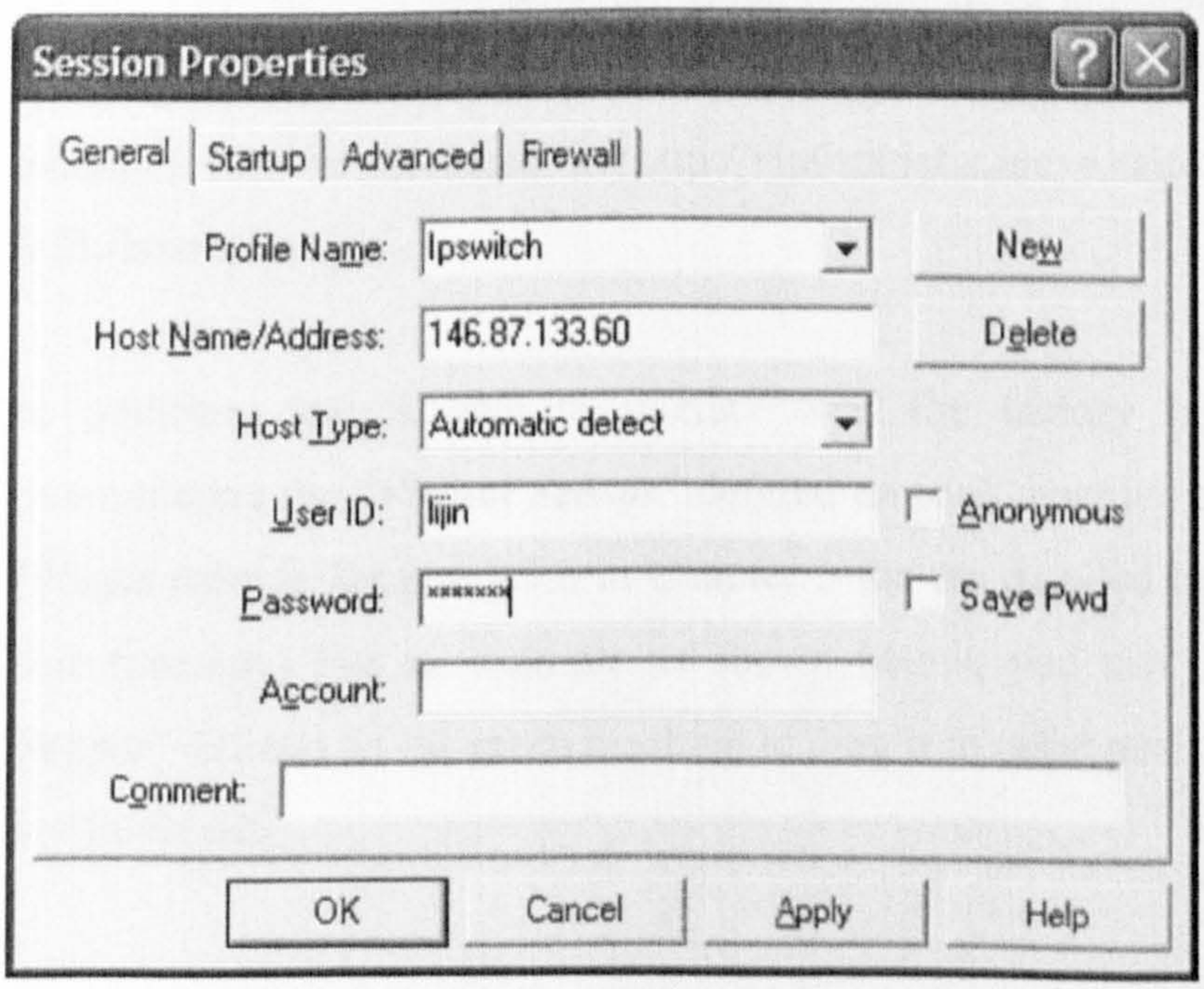
the initial rotation angle is 0° for rotation group.

3. Distributing the virtual scenes across the Internet

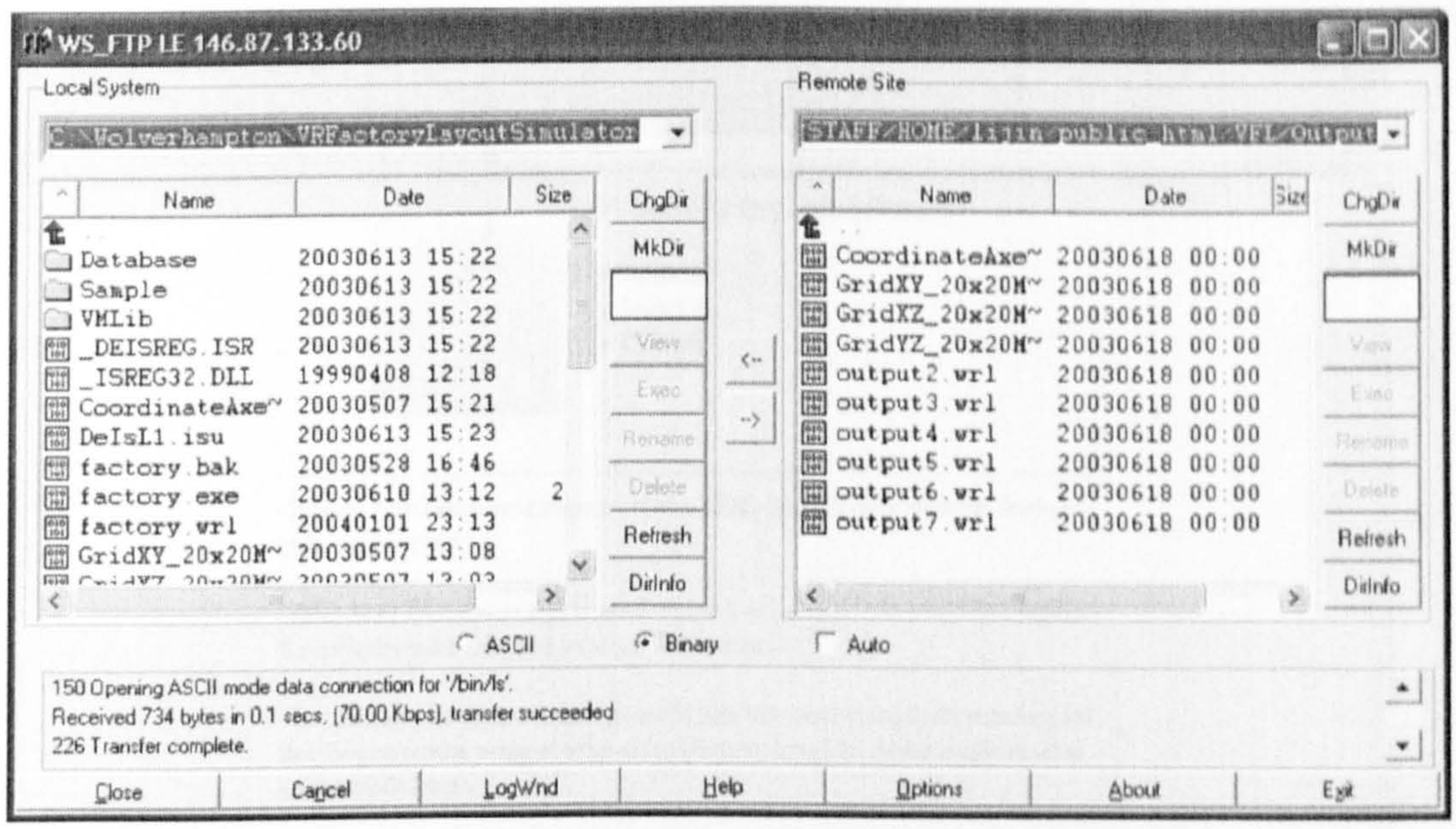
The section presents how to distribute the VRML-based factory layout scenes produced by the VFLS software toolkit across the Internet in order to allow multiple users located in the different places to remotely access, evaluate, interactive with the virtual layout scenes in real time through the WWW.

I. Uploading the virtual factory scenes onto the Web server

After you saved a factory layout scene in the VFLS toolkit, the VRML-based layout senses along with the VMD can be uploaded onto the Web server via the common FTP program such as WS_FTP. The WS_FTP can be freely downloaded from the Internet for the academic use purpose. You can use the free FTP program to logon a Web-sever and send the factory layout file remotely to your Web space on sever. In this project, the author used the server of University of Salford* to carry out this experiment as shown as below.



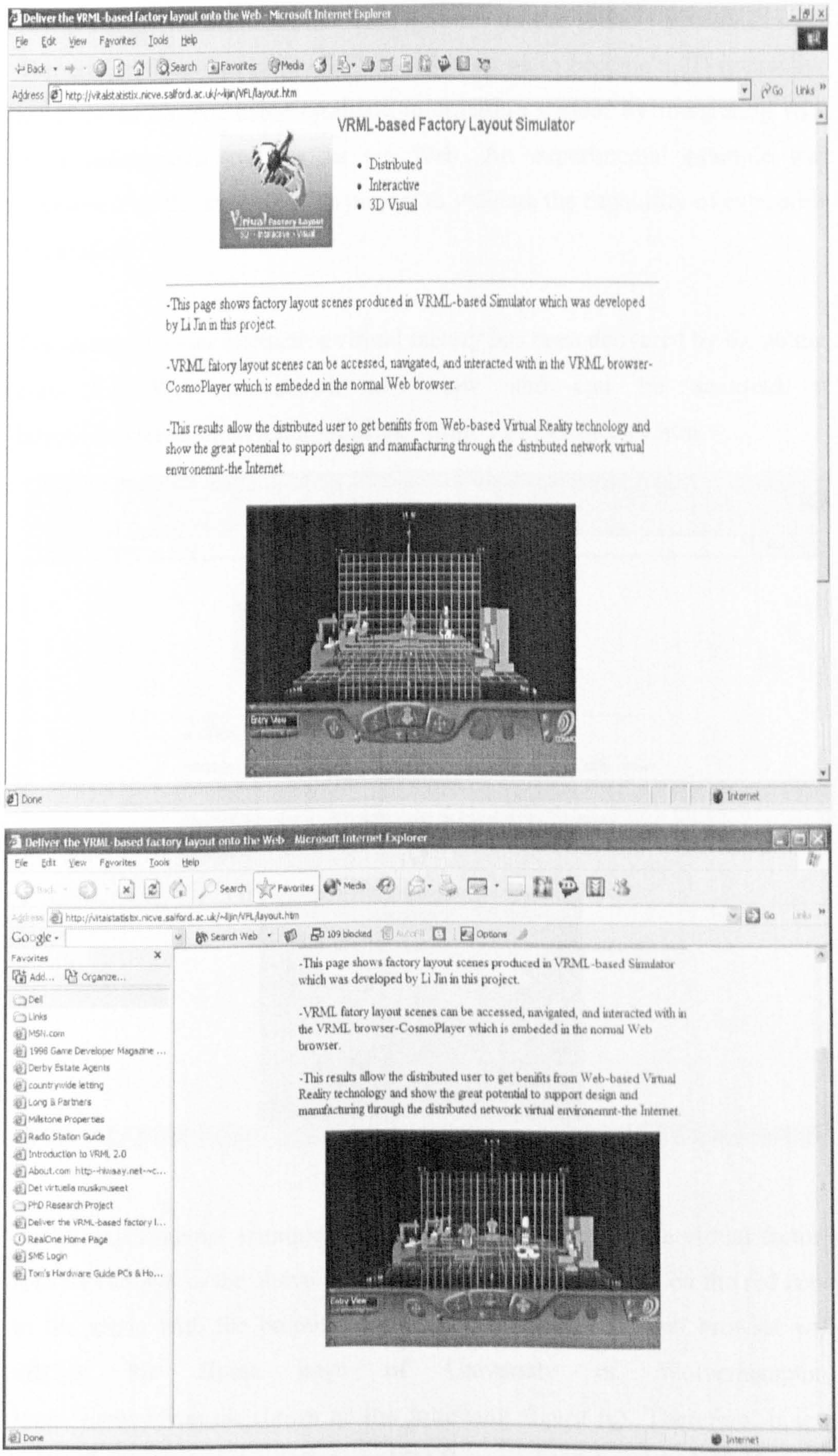
* The author worked as a Research Fellow in the Centre for Virtual Environments, University of Salford from 2002 to 2003.



II. Remotely access and interaction with the factory layout scenes in virtual environment on the Web

After you uploaded the VRML-based layout senses produced in VFLS onto the Web server via the common FTP program, it then can be displayed by a VRML browser — CosmoPlayer, which is embedded in normal HTML-based Webpages. The author carried out this experiment as below. The experimental example can be accessed at http://vitalstatistix.nicve.salford.ac.uk/ ~lijin/ VFL/layout.htm.

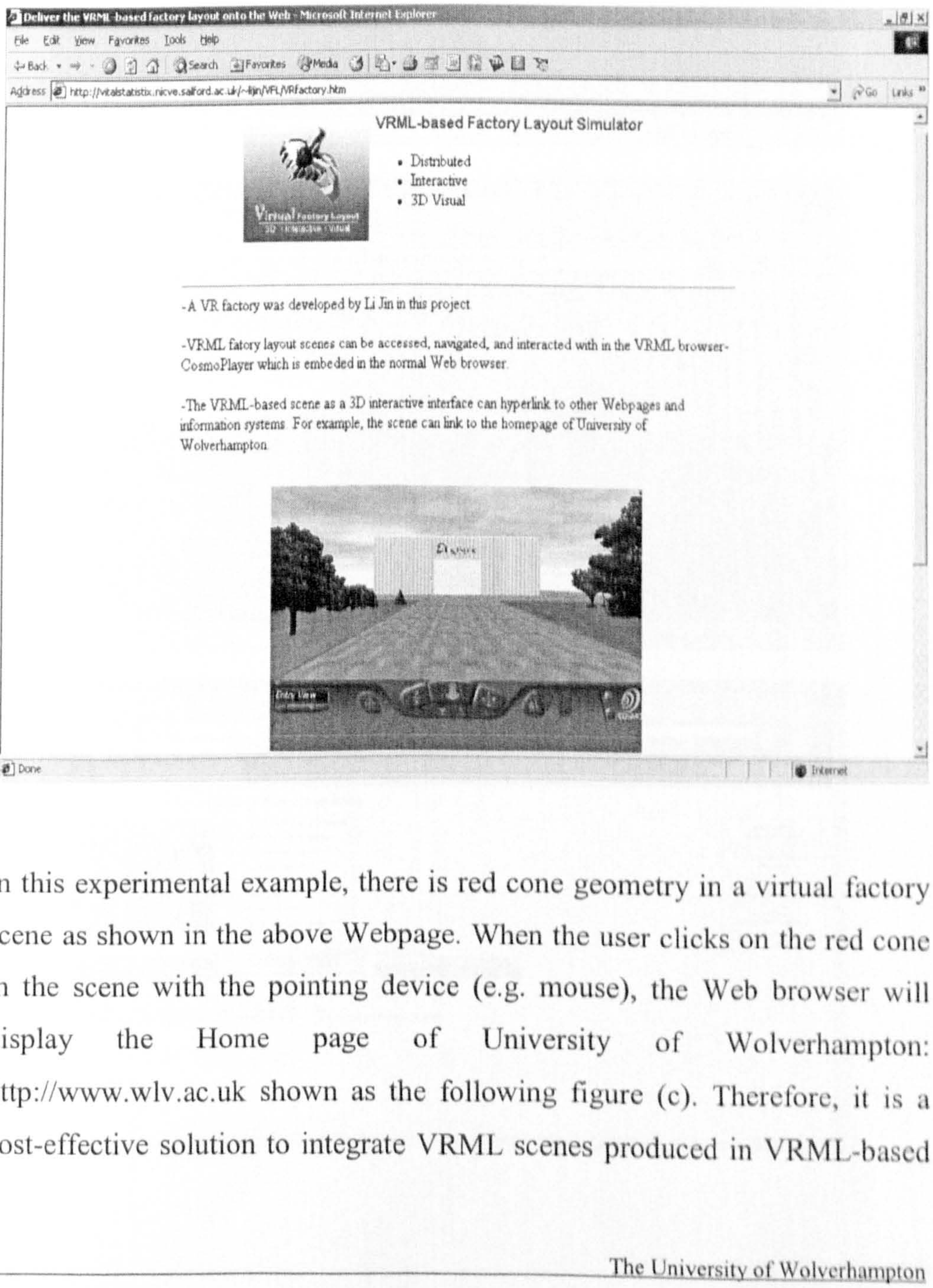
In addition, you is able to interact with the factory layout scenes by manipulating the "***Virtual Sensor***" defined on each machine on the Webpage. (Please refer to Section 5.3.6 in Chapter 5 for the detailed information about this function.) For an example as shown below, you can use the "***Virtual Sensor***" defined on the green machine to drag it to other position in the virtual environment.


Deliver the VRML-based factory layout onto the Web - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://vitalstatistix.nicve.salford.ac.uk/~lijin/VFL/layout.htm
Go Links
VRML-based Factory Layout Simulator
Virtual Factory Layout
• Distributed
• Interactive
• 3D Visual
-This page shows factory layout scenes produced in VRML-based Simulator which was developed by Li Jin in this project.
-VRML fatory layout scenes can be accessed, navigated, and interacted with in the VRML browser-CosmoPlayer which is embeded in the normal Web browser.
-This results allow the distributed user to get benifits from Web-based Virtual Reality technology and show the great potential to support design and manufacturing through the distributed network virtual environemnt-the Internet.
Entry View
Done
Internet
Deliver the VRML-based factory layout onto the Web - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Search Favorites Media
Address http://vitalstatistix.nicve.salford.ac.uk/~lijin/VFL/layout.htm
Go
Google
Search Web
109 blocked
Options
Favorites
Add... Organize...
Dell
Links
MSN.com
1998 Game Developer Magazine ...
Derby Estate Agents
countrywide letting
Long & Partners
Millstone Properties
Radio Station Guide
Introduction to VRML 2.0
About.com http://iwaay.net/~c...
Det virtuella musikmuseet
PhD Research Project
Deliver the VRML-based factory l...
RealOne Home Page
SMS Login
Tom's Hardware Guide PCs & Ho...
-This page shows factory layout scenes produced in VRML-based Simulator which was developed by Li Jin in this project.
-VRML fatory layout scenes can be accessed, navigated, and interacted with in the VRML browser-CosmoPlayer which is embeded in the normal Web browser.
-This results allow the distributed user to get benifits from Web-based Virtual Reality technology and show the great potential to support design and manufacturing through the distributed network virtual environemnt-the Internet.
Entry View
Done
Internet

III. Integrating with other information systems on the Web

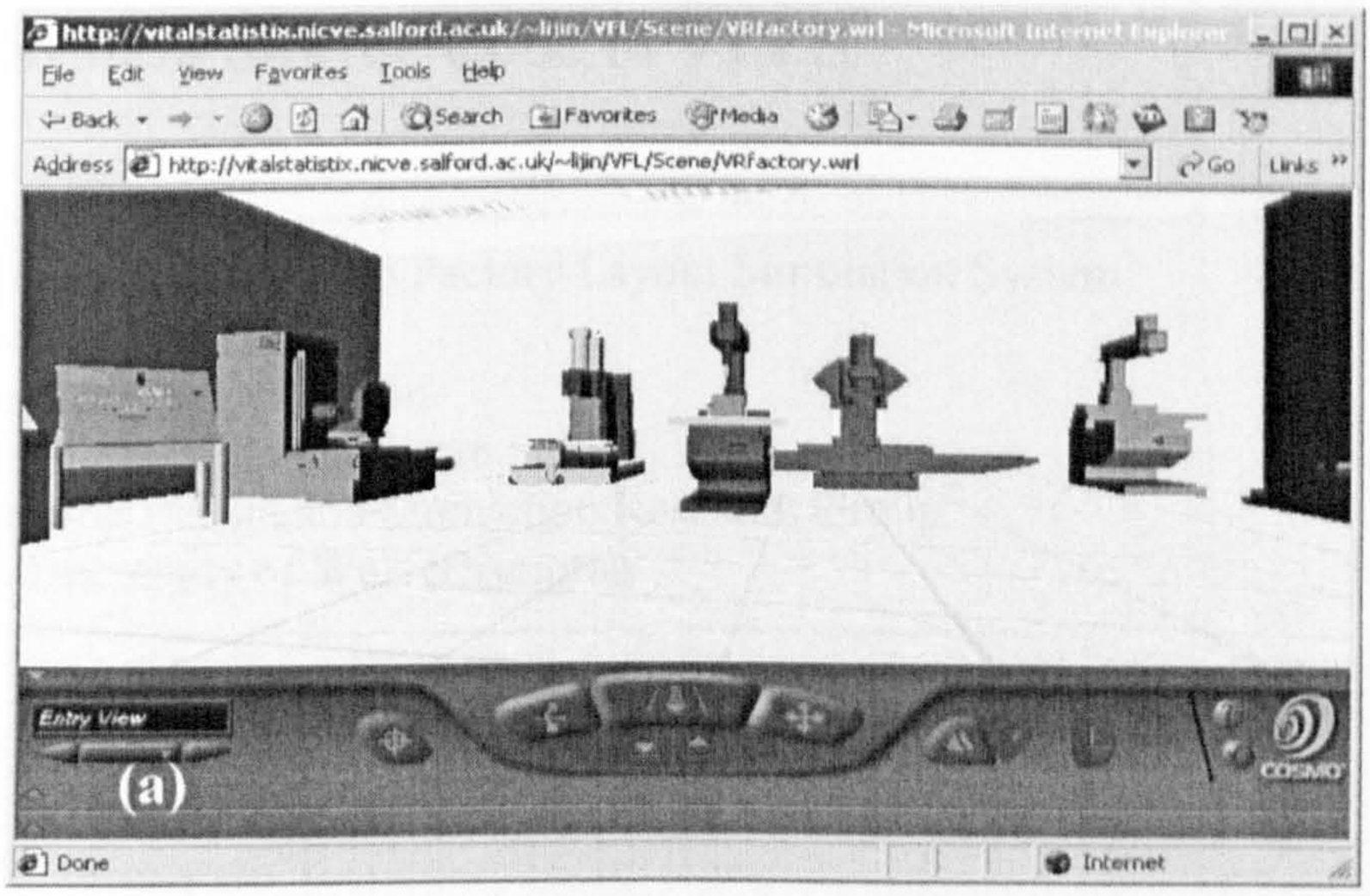
The section describes how a VRML-based scene to become a 3D interactive interface of an extensible synthetic information system by integrating with other information systems on the Web. An experimental example was developed by the author in this project to validate the capability of extensible integration.

The example — an interactive virtual factory has been delivered by the author onto the Web as shown as below and can be accessed at http://vitalstatistix.nicve.salford.ac.uk/~lijin/ VFL/VRfactory.htm.

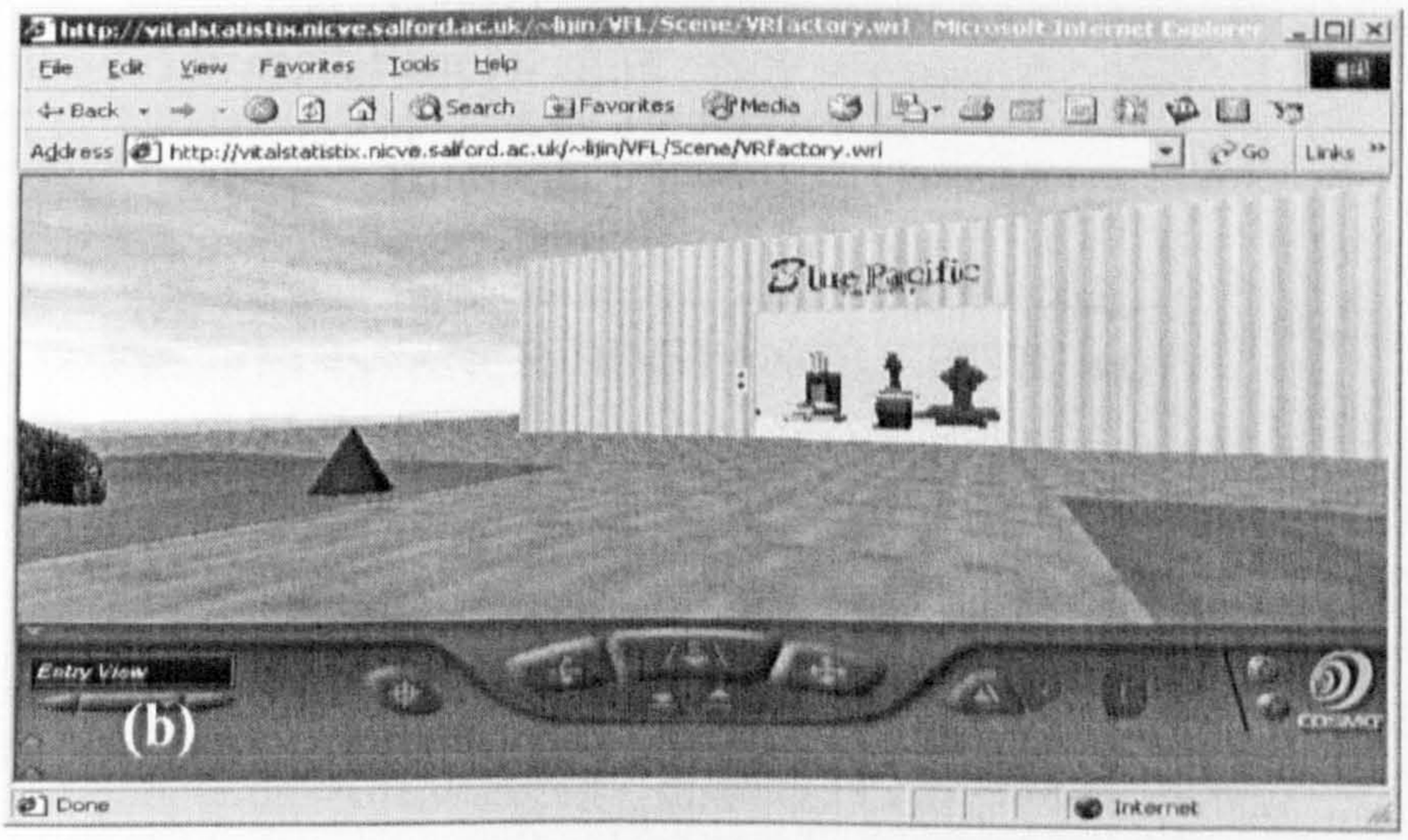


In this experimental example, there is red cone geometry in a virtual factory scene as shown in the above Webpage. When the user clicks on the red cone in the scene with the pointing device (e.g. mouse), the Web browser will display the Home page of University of Wolverhampton: http://www.wlv.ac.uk shown as the following figure (c). Therefore, it is a cost-effective solution to integrate VRML scenes produced in VRML-based

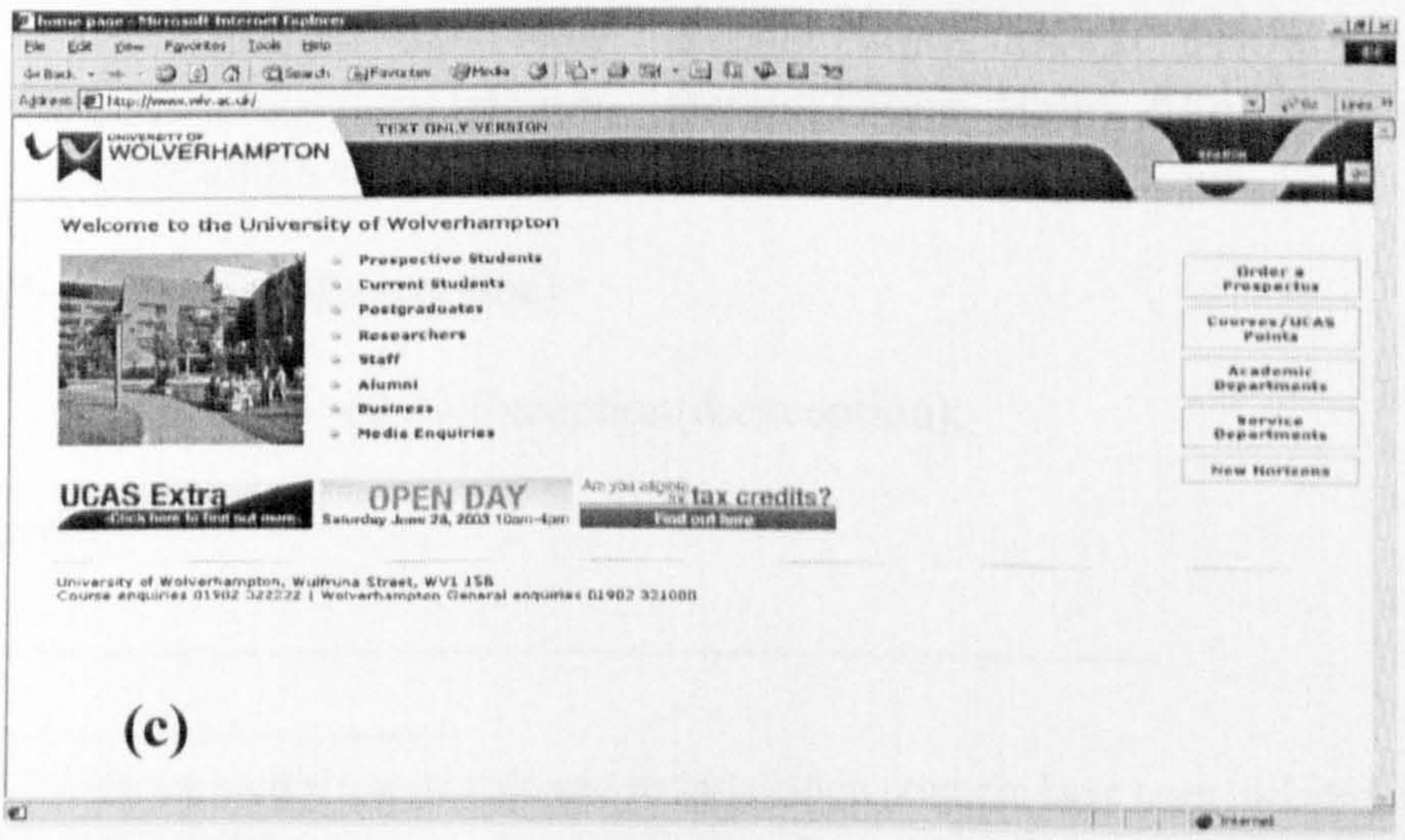
Factory Layout Simulator (VFLS) with other HTML-based files (based on text, image, and video files) and even database and information system through the Web.



(a)



(b)



(c)

Appendix B

Core System Source Code of VFLS*

```
//---------------------------------------------------------------------------
// Distributed VRML-based Factory Layout Simulation System
// File name: factory.cpp
// Software Developer: Li Jin
// Contact: jinhull@hotmail.com
// Engineering Design and Simulation Research Group
// SEBE, University of Wolverhampton
// ---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
USERES("factory.res");
USEFORM("layout.cpp", Form1);
USEFORM("View.cpp", Form2);
USEFORM("ModelNav.cpp", Form_ModelView);
USEFORM("About.cpp", Form_About);
USEFORM("Help.cpp", Form_Help);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
      try
      {
            Application->Initialize();
            Application->CreateForm(__classid(TForm1), &Form1);
            Application->CreateForm(__classid(TForm2), &Form2);
            Application->CreateForm(__classid(TForm_ModelView),
&Form_ModelView);
            Application->CreateForm(__classid(TForm_About), &Form_About);
            Application->CreateForm(__classid(TForm_Help), &Form_Help);
            Application->Run();
      }
      catch (Exception &exception)
      {
            Application->ShowException(&exception);
      }
      return 0;
}
//---------------------------------------------------------------------------
```

* The VFLS software toolkit source code and its installation program have been included in the CD attached to this thesis.

```
//---------------------------------------------------------------------------
// Distributed VRML-based Factory Layout Simulation System
// File name: layout.cpp
// Software Developer: Li Jin
// Contact: jinhull@hotmail.com
// Engineering Design and Simulation Research Group
// SEBE, University of Wolverhampton
// ---------------------------------------------------------------------------
#include <vcl.h>
#include <fstream.h>
#include <iostream.h>
#pragma hdrstop

#include "layout.h"
#include "ModelNav.h"
#include "View.h"
#include "About.h"
#include "Help.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "SHDocVw_OCX"
#pragma resource "*.dfm"

AnsiString ProjectCurrentDir;
Double PI = 3.1415926;


TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

//---------------------------------------------------------------------------
// Main Interface
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 AnsiString Logofile, CurrentWorld;

 ProjectCurrentDir = GetCurrentDir();
 Logofile = ProjectCurrentDir + "\\" + "logo.bmp";
 //Form1->DBImage1->Picture->LoadFromFile("C:/Program
Files/Borland/CBuilder5/Wolver/Photo/logo.bmp");
 Form1->DBImage1->Picture->LoadFromFile(Logofile);
```

```
 //Load inital virtual world
 CurrentWorld = ProjectCurrentDir + "\\" + "factory.wrl";
 TVariant vAddress = {CurrentWorld};
 Form1->CppWebBrowser2->Navigate2(&vAddress);

}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//Catagory selection
void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
 AnsiString Filterstr;

 AnsiString ModelPath = ProjectCurrentDir + "\\VMLib\\";

 Form1->Table_VM->Close();
 Form1->Query_VM->Close();

 if (!Form1->Query_VM->Prepared)
    Form1->Query_VM->Prepare();
 Filterstr = Form1->ComboBox1->Text;
 //---------------------------------------------------------------------------
 //According to combox text item set up the filter of Query
 Form1->Query_VM->Filter= "Category='" + Filterstr + "'";
 Form1->DBGrid1->DataSource = Form1->DataSource_VMFilter;
 Form1->DBImage1->DataField = "Photo";
 Form1->DBImage1->DataSource = Form1->DataSource_VMFilter;

 //Form1->DBMemo1->DataField = "Description";
 //Form1->DBMemo1->DataSource = Form1->DataSource_VMFilter;

 //Form1->DBRichEdit1->DataField = "Description";
 //Form1->DBRichEdit1->DataSource = Form1->DataSource_VMFilter;

 Form1->Query_VM->Open();
 Form1->Table_VM->Open();

  //enable button "View Model"
 Form1->Button_ViewModel->Enabled = true;


 //-------------------------
 Form1->Label_ModelPath->Caption = ModelPath;
```

```
//once a model has been selected, enable manipulation functions
Form1->CheckBox_Move->Enabled = true;
Form1->CheckBox_Scale->Enabled = true;
Form1->CheckBox_Rotation ->Enabled = true;

//enable "Reset" button
Form1->Button_Reset->Enabled = true;

//Enable "Add" button
 Form1->Button_Add->Enabled = true;


}

//---------------------------------------------------------------------
// 3D model navigation and viewing
void __fastcall TForm1::Button_ViewModelClick(TObject *Sender)
{


 //-------------------------------------
 //Call models in local disk
 //char *str = "G:/VRML/Engine/ENGINE_AN.WRL";
 //AnsiString Modelname = "/VMLib/Model3-3.wrl";
 AnsiString ModelAdress, ModelName;
 AnsiString ModelPath = ProjectCurrentDir + "\\VMLib\\";

 ModelName = Form1->DBGrid1->SelectedField->Text + ".wrl";
 ModelAdress = ModelPath + ModelName;


 //ModelAdress = ProjectCurrentDir + Modelname;

 //TVariant vAddress = {str}
 TVariant vAddress = {ModelAdress};
 //TVariant vAddress = {"G:/VRML/Engine/ENGINE_AN.WRL"};
 //Form2->Show();

 //-------------------------------------
 //call models in remote sever
 //Form1->CppWebBrowser1->Navigate2(WideString(L"http://pers-
www.wlv.ac.uk/~in6716/current/Hulluniversity.wrl"));

 // Form1->CppWebBrowser1->Navigate2(&vAddress);

 //-------------------------------------
```

```
//pass modelname to manipulation panel
Form1->Label_ModelPath->Caption = ModelPath;


//---------------------------------------
//once a model has been selected, enable manipulation functions
//Form1->CheckBox_Move->Enabled = true;
//Form1->CheckBox_Scale->Enabled = true;
//Form1->CheckBox_Rotation ->Enabled = true;

  Form_ModelView->Show();
  Form_ModelView->CppWebBrowser1->Navigate2(&vAddress);



}
//---------------------------------------------------------
//Turn on/off "Move" function
void __fastcall TForm1::CheckBox_MoveClick(TObject *Sender)
{
  //---------------------------------
  //Set Move enabled
  if (Form1->CheckBox_Move->Checked == true)
     {Form1->Label_MX->Enabled = true;
     Form1->Edit_MX->Enabled = true;
     Form1->UpDown_MX->Enabled = true;

     Form1->Label_MY->Enabled = true;
     Form1->Edit_MY->Enabled = true;
     Form1->UpDown_MY->Enabled = true;

     Form1->Label_MZ->Enabled = true;
     Form1->Edit_MZ->Enabled = true;
     Form1->UpDown_MZ->Enabled = true;
     }
     else {
     Form1->Label_MX->Enabled = false;
     Form1->Edit_MX->Enabled = false;
     Form1->UpDown_MX->Enabled = false;

     Form1->Label_MY->Enabled = false;
     Form1->Edit_MY->Enabled = false;
     Form1->UpDown_MY->Enabled = false;

     Form1->Label_MZ->Enabled = false;
     Form1->Edit_MZ->Enabled = false;
     Form1->UpDown_MZ->Enabled = false;
     }
```

```
}
//---------------------------------------------------------------------------
////Turn on/off "Scale" function
void __fastcall TForm1::CheckBox_ScaleClick(TObject *Sender)
{
  //---------------------------------
  //Set Scale enabled
  if (Form1->CheckBox_Scale->Checked == true)
     {Form1->Label_SX->Enabled = true;
     Form1->Edit_SX->Enabled = true;
     Form1->UpDown_SX->Enabled = true;

     Form1->Label_SY->Enabled = true;
     Form1->Edit_SY->Enabled = true;
     Form1->UpDown_SY->Enabled = true;

     Form1->Label_SZ->Enabled = true;
     Form1->Edit_SZ->Enabled = true;
     Form1->UpDown_SZ->Enabled = true;
     }
     else {
     Form1->Label_SX->Enabled = false;
     Form1->Edit_SX->Enabled = false;
     Form1->UpDown_SX->Enabled = false;

     Form1->Label_SY->Enabled = false;
     Form1->Edit_SY->Enabled = false;
     Form1->UpDown_SY->Enabled = false;

     Form1->Label_SZ->Enabled = false;
     Form1->Edit_SZ->Enabled = false;
     Form1->UpDown_SZ->Enabled = false;
     }
}
//---------------------------------------------------------------------------
//Turn on/off "Rotation" function
void __fastcall TForm1::CheckBox_RotationClick(TObject *Sender)
{
  //---------------------------------
  //Set Rotation function enabled
  if (Form1->CheckBox_Rotation->Checked == true)
     {Form1->RadioButton_X->Enabled = true;
     Form1->RadioButton_Y->Enabled = true;
     Form1->RadioButton_Z->Enabled = true;

     Form1->Label_Angle->Enabled = true;
     Form1->Edit_Angle->Enabled = true;
     Form1->UpDown_Angle->Enabled = true;
```

```
}
else {
Form1->RadioButton_X->Enabled = false;
Form1->RadioButton_Y->Enabled = false;
Form1->RadioButton_Z->Enabled = false;

Form1->Label_Angle->Enabled = false;
Form1->Edit_Angle->Enabled = false;
Form1->UpDown_Angle->Enabled = false;
}
}
//---------------------------------------------------------------------
// Click UpDown button to adjust the value of tranlation
//---------------------------------------------------------------------
void __fastcall TForm1::UpDown_MXClick(TObject *Sender, TUDBtnType
Button)
{
  Form1->UpDown_MX->Position = Form1->Edit_MX->Text.ToDouble();
  Form1->Edit_MX->Text = Form1->UpDown_MX->Position;
}
//---------------------------------------------------------------------

void __fastcall TForm1::UpDown_MYClick(TObject *Sender, TUDBtnType
Button)
{
  Form1->UpDown_MY->Position = Form1->Edit_MY->Text.ToDouble();
  Form1->Edit_MY->Text = Form1->UpDown_MY->Position;
}
//---------------------------------------------------------------------

void __fastcall TForm1::UpDown_MZClick(TObject *Sender, TUDBtnType
Button)
{
  Form1->UpDown_MZ->Position = Form1->Edit_MZ->Text.ToDouble();
  Form1->Edit_MZ->Text = Form1->UpDown_MZ->Position;
}

//---------------------------------------------------------------------
// Input value to translation
//-------------------------

void __fastcall TForm1::Edit_MXChange(TObject *Sender)
{
  Form1->UpDown_MX->Position = Form1->Edit_MX->Text.ToDouble();
}
//---------------------------------------------------------------------

void __fastcall TForm1::Edit_MYChange(TObject *Sender)
```

```
{
  Form1->UpDown_MY->Position = Form1->Edit_MY->Text.ToDouble();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Edit_MZChange(TObject *Sender)
{
  Form1->UpDown_MZ->Position = Form1->Edit_MZ->Text.ToDouble();
}

//---------------------------------------------------------------------------
// Click UpDown button to adjust the value of Scale along X axis
//---------------------------------------------------------------------------
void __fastcall TForm1::UpDown_SXClick(TObject *Sender, TUDBtnType
Button)
{
  Double InputVal, temp, OutputVal;
  Double static preInputVal=0.0;
   AnsiString OutText;

  InputVal = Form1->Edit_SX->Text.ToDouble();

preInputVal=InputVal;

//Form1->Label1->Caption = AnsiString(Form1->UpDown_SX->Position);

//Form1->UpDown_SX->Position = Form1->Edit_SX->Text.ToDouble();
if (InputVal < 1.0)
{
 // temp = InputVal * 10;
    //Form1->UpDown_SX->Position = temp;
 OutputVal =  ((Double) Form1->UpDown_SX->Position)/10.0;
//  Form1->Edit_SX->Text = OutputVal;
  }

// if ((InputVal ==1.0)|| (InputVal >1.0))
   if (InputVal >=1.0)
 {
  //Form1->UpDown_SX->Position = InputVal;
  //OutputVal =  ((Double) Form1->UpDown_SX->Position)/10.0;
   OutputVal= Form1->UpDown_SX->Position;
 // Form1->Edit_SX->Text = Form1->UpDown_SX->Position;

}
```

```
if(Form1->UpDown_SX->Position==0 && preInputVal==1.0 )
{
 Form1->UpDown_SX->Position=Form1->UpDown_SX->Position+9;
 OutputVal= 0.9;
 }

 /*  the program do the following atumatically, strange...
 if(Form1->UpDown_SX->Position==10 && preInputVal==0.9 )
{
    Form1->UpDown_SX->Position=Form1->UpDown_SX->Position-9;
    OutputVal= 1;
   }
    */
    Form1->Edit_SX->Text = OutputVal;

}
//---------------------------------------------------------------------------
// input Scale value for X
void __fastcall TForm1::Edit_SXChange(TObject *Sender)
{

  Double InputVal;

  InputVal = Form1->Edit_SX->Text.ToDouble();

   if (InputVal < 1.0)
   {
    InputVal = InputVal * 10;
    Form1->UpDown_SX->Position = InputVal;
   }

   if ((InputVal ==1.0)||(InputVal >1.0))
   {
    Form1->UpDown_SX->Position = InputVal;
   }

}
//---------------------------------------------------------
//Click UpDown button to adjust the value of Scale along Y axis
//---------------------------------------------------------
void __fastcall TForm1::UpDown_SYClick(TObject *Sender, TUDBtnType
Button)
{
  Double InputVal, temp, OutputVal;
  Double static preInputVal=0.0;
  AnsiString OutText;
```

```
InputVal = Form1->Edit_SY->Text.ToDouble();

preInputVal=InputVal;

//Form1->Label1->Caption = AnsiString(Form1->UpDown_SY->Position);

if (InputVal < 1.0)
{
 OutputVal =  ((Double) Form1->UpDown_SY->Position)/10.0;
  }

if (InputVal >=1.0)
{
 OutputVal= Form1->UpDown_SY->Position;
  }

if(Form1->UpDown_SY->Position==0 && preInputVal==1.0 )
{
 Form1->UpDown_SY->Position=Form1->UpDown_SY->Position+9;
 OutputVal=  0.9;
 }

Form1->Edit_SY->Text = OutputVal;
}
//---------------------------------------------
// input scale value for Y
void __fastcall TForm1::Edit_SYChange(TObject *Sender)
{
  Double InputVal;

  InputVal = Form1->Edit_SY->Text.ToDouble();

  if (InputVal < 1.0)
  {
   InputVal = InputVal * 10;
   Form1->UpDown_SY->Position = InputVal;
  }

  if ((InputVal ==1.0)||(InputVal >1.0))
  {
   Form1->UpDown_SY->Position = InputVal;
  }

}
//-------------------------------------------------------------
//Click UpDown button to adjust the value of Scale along Z axis
```

```
//---------------------------------------------------
void __fastcall TForm1::UpDown_SZClick(TObject *Sender, TUDBtnType
Button)
{
 Double InputVal, temp, OutputVal;
  Double static preInputVal=0.0;
  AnsiString OutText;

  InputVal = Form1->Edit_SZ->Text.ToDouble();

  preInputVal=InputVal;

  //Form1->Label1->Caption = AnsiString(Form1->UpDown_SZ->Position);

  if (InputVal < 1.0)
  {
   OutputVal =  ((Double) Form1->UpDown_SZ->Position)/10.0;

   }

  if (InputVal >=1.0)
  {
   OutputVal= Form1->UpDown_SZ->Position;
   }

    if(Form1->UpDown_SZ->Position==0 && preInputVal==1.0 )
    {
     Form1->UpDown_SZ->Position=Form1->UpDown_SZ->Position+9;
     OutputVal=  0.9;
     }

    Form1->Edit_SZ->Text = OutputVal;

}
//---------------------------------------------------
//  input scale value for Z
void __fastcall TForm1::Edit_SZChange(TObject *Sender)
{
  Double InputVal;

  InputVal = Form1->Edit_SZ->Text.ToDouble();

  if (InputVal < 1.0)
  {
   InputVal = InputVal * 10;
   Form1->UpDown_SZ->Position = InputVal;
  }
```

```
  if ((InputVal ==1.0)||(InputVal >1.0))
  {
   Form1->UpDown_SZ->Position = InputVal;
  }
}
//---------------------------------------------------
//Adjust angle of Rotation
//---------------------------------------------------------------------

void __fastcall TForm1::UpDown_AngleClick(TObject *Sender,
    TUDBtnType Button)
{
  Form1->UpDown_Angle->Position = Form1->Edit_Angle->Text.ToDouble();
  Form1->Edit_Angle->Text = Form1->UpDown_Angle->Position;
}
//---------------------------------------------------------------------

void __fastcall TForm1::Edit_AngleChange(TObject *Sender)
{
  Form1->UpDown_Angle->Position = Form1->Edit_Angle->Text.ToDouble();
}
//---------------------------------------------------------------------
// Add model
void __fastcall TForm1::Button_AddClick(TObject *Sender)
{
//-----------------------------------
//Add the code here to merge two wrl files
AnsiString ModelName, ModelPath, ModelAdress, CurrentWorld;
AnsiString TranStr, ScaleStr, RotationStr;
AnsiString X_axis = " 1 0 0 ", Y_axis = " 0 1 0 ", Z_axis = " 0 0 1 ";
double Rot_radian;
AnsiString Rot_radianStr;
FILE *fout_C;

//ModelName = Form1->Label_ModelName->Caption;
ModelName = Form1->DBText_ModelName->Field->DisplayText + ".wrl";
ModelPath = Form1->Label_ModelPath->Caption;
ModelAdress = ModelPath + ModelName;
CurrentWorld = ProjectCurrentDir + "\\" + "factory.wrl";
  //Form1->Label1->Caption = CurrentWorld;
  Form1->Label_Infor->Caption = "The current virtual world is: " +
CurrentWorld;

  fout_C = fopen ( CurrentWorld.c_str(), "a+");
  //fout_C = fopen ( "C:\\Phd-Thesis\\Wolver\\factory.wrl", "a+");
  fprintf(fout_C, "%s", "#Add new machine\n" );
  fprintf(fout_C, "%s", "Group{children[Transform{\n");
```

```
//----------------
//Translation:
TranStr = "translation " + Form1->Edit_MX->Text
        + " " + Form1->Edit_MY->Text
        + " " + Form1->Edit_MZ->Text
        + "\n";

//----------------
//Scale:
ScaleStr = "scale " + Form1->Edit_SX->Text
        + " " + Form1->Edit_SY->Text
        + " " + Form1->Edit_SZ->Text
        + "\n";

//----------------
//Rotation:
Rot_radian = ((Form1->Edit_Angle->Text.ToDouble())* PI)/180.0;
Rot_radianStr = AnsiString (Rot_radian);

//Rotate around Y axis
if(Form1->RadioButton_Y->Checked == true)
  RotationStr = "rotation " + Y_axis + Rot_radianStr + "\n";

if(Form1->RadioButton_X->Checked == true)
  RotationStr = "rotation " + X_axis + Rot_radianStr + "\n";

if(Form1->RadioButton_Z->Checked == true)
  RotationStr = "rotation " + Z_axis + Rot_radianStr + "\n";

fprintf (fout_C, "%s", TranStr.c_str());
fprintf (fout_C, "%s", ScaleStr.c_str());
fprintf (fout_C, "%s", RotationStr.c_str());


//--------------------
fprintf (fout_C, "%s", "children Inline{\n");
fprintf (fout_C, "%s", "url \"");
fprintf (fout_C, "%s", ModelPath.c_str());
fprintf (fout_C, "%s", ModelName.c_str());
fprintf (fout_C, "%s", "\"\n");
fprintf (fout_C, "%s", "}\n");
fprintf (fout_C, "%s", "},]}\n\n");

fclose (fout_C);

//----------------------
// view updated world
```

```
  TVariant vAddress = {CurrentWorld};
  Form1->CppWebBrowser2->Navigate2(&vAddress);

  //-----------------------------
  // Enable "Save As" button
  Form1->Button_Save->Enabled = true;
}
//---------------------------------------------------------------------------
// Reset world
//--------------------
void __fastcall TForm1::Button_ClearClick(TObject *Sender)
{
 //-----------------------------------
  //Add the code here to overwrite a wrl files

  FILE *fout_C,  *fout_BK;
  AnsiString BackupWorld, CurrentWorld, ResetWorld;

  BackupWorld = ProjectCurrentDir + "\\" + "factory.bak";
  CurrentWorld = ProjectCurrentDir + "\\" + "factory.wrl";
 // Form1->Label1->Caption = CurrentWorld;

  fout_BK = fopen ( BackupWorld.c_str(), "r");
  fout_C = fopen (CurrentWorld.c_str(), "w");
  do
 {fputc(fgetc(fout_BK), fout_C);}
 while (!feof(fout_BK));

    //fprintf(fout_BK, "%s", fout_C);

 fclose (fout_C);
 fclose (fout_BK);

 //----------------------
 // view reseted world
 TVariant vAddress = {CurrentWorld};
 Form1->CppWebBrowser2->Navigate2(&vAddress);

 //-----------------------
   // Disenable the Button "Save As"
   Form1->Button_Save->Enabled = false;
}

//---------------------------------------------------------------------------
// To save a layout file
void __fastcall TForm1::Button_SaveClick(TObject *Sender)
{
  /*
```

```
//-------------------------------------
//Call models in local disk
char *str = "C:/Phd-Thesis/Wolver/VMLib/Model1-3.wrl";
TVariant vAddress = {str};
//TVariant vAddress = {"G:/VRML/Engine/ENGINE_AN.WRL"};
//Form2->Show();

//-------------------------------------
//call models in remote sever
//Form1->CppWebBrowser2->Navigate2(WideString(L"http://pers-
www.wlv.ac.uk/~in6716/current/Hulluniversity.wrl"));

 Form1->CppWebBrowser2->Navigate2(&vAddress);
 */

 FILE *fout_C,  *fout_BK;
 AnsiString CurrentWorld, NewNameWorld;


Form1->SaveDialog1->InitialDir =  ProjectCurrentDir + "\\Output\\";
Form1->SaveDialog1->Filter ="VRML Files(*.wrl)|*.wrl";

Form1->SaveDialog1->DefaultExt="wrl";

if (Form1->SaveDialog1->Execute());
{
NewNameWorld = Form1->SaveDialog1->FileName;
CurrentWorld = ProjectCurrentDir + "\\" + "factory.wrl";

fout_BK = fopen ( CurrentWorld.c_str(), "r");
fout_C = fopen (NewNameWorld.c_str(), "w");
do
{fputc(fgetc(fout_BK), fout_C);}
while (!feof(fout_BK));


fclose (fout_C);
fclose (fout_BK);

  }

}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::Button_AboutClick(TObject *Sender)
{
 Form_About->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button_QuitClick(TObject *Sender)
{
 Form1->Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button_ResetClick(TObject *Sender)
{
   Form1->CheckBox_Move->Checked = false;
   Form1->CheckBox_Scale->Checked = false;
   Form1->CheckBox_Rotation->Checked = false;

   Form1->Edit_MX->Text = "0";
   Form1->Edit_MY->Text = "0";
   Form1->Edit_MZ->Text = "0";

   Form1->Edit_SX->Text = "1";
   Form1->Edit_SY->Text = "1";
   Form1->Edit_SZ->Text = "1";

   Form1->RadioButton_X->Checked = false;
   Form1->RadioButton_Y->Checked = false;
   Form1->RadioButton_Z->Checked = false;
   Form1->Edit_Angle->Text = "0";

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button_HelpClick(TObject *Sender)
{
 Form_Help->Show();
}
```

```
//---------------------------------------------------------------------------
// Distributed VRML-based Factory Layout Simulation System
// File name: ModelNav.cpp
// Software Developer: Li Jin
// Contact: jinhull@hotmail.com
// Engineering Design and Simulation Research Group
// SEBE, University of Wolverhampton
// ---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "ModelNav.h"
#include "layout.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "SHDocVw_OCX"
#pragma resource "*.dfm"
TForm_ModelView *Form_ModelView;
//---------------------------------------------------------------------------
__fastcall TForm_ModelView::TForm_ModelView(TComponent* Owner)
        : TForm(Owner)
{
}

void __fastcall TForm_ModelView::Button_CloseClick(TObject *Sender)
{
 Form_ModelView->Hide();
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------
// Distributed VRML-based Factory Layout Simulation System
// File name: About.cpp
// Software Developer: Li Jin
// Contact: jinhull@hotmail.com
// Engineering Design and Simulation Research Group
// SEBE, University of Wolverhampton
// ---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "About.h"
#include "layout.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_About *Form_About;
//---------------------------------------------------------------------------
__fastcall TForm_About::TForm_About(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm_About::Button_ExitClick(TObject *Sender)
{
 Form_About->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TForm_About::FormCreate(TObject *Sender)
{

}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------
// Distributed VRML-based Factory Layout Simulation System
// File name: Help.cpp
// Software Developer: Li Jin
// Contact: jinhull@hotmail.com
// Engineering Design and Simulation Research Group
// SEBE, University of Wolverhampton
// ---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Help.h"
#include "layout.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_Help *Form_Help;
//---------------------------------------------------------------------------
__fastcall TForm_Help::TForm_Help(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm_Help::Button_CloseClick(TObject *Sender)
{
  Form_Help->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TForm_Help::FormCreate(TObject *Sender)
{
  AnsiString HelpFileName;

  HelpFileName = GetCurrentDir() + "\\help.rtf";
  Form_Help->RichEdit_Help->Lines->LoadFromFile(HelpFileName);
}
//---------------------------------------------------------------------------
```